VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

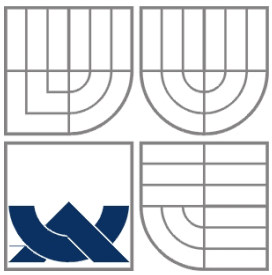
NÁVRH A REALIZACE PROTOTYPU ZAŘÍZENÍ PRO VÍCEKANÁLOVÉ SLEDOVÁNÍ MĚŘENÝCH VELIČIN PO USB

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

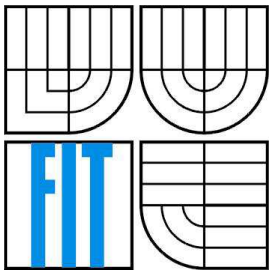
AUTOR PRÁCE
AUTHOR

JAN NEVORAL

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NÁVRH A REALIZACE PROTOTYPU ZAŘÍZENÍ PRO VÍCEKANÁLOVÉ SLEDOVÁNÍ MĚŘENÝCH VELIČIN PO USB

DESIGN AND REALIZATION OF DEVICE-PROTOTYPE FOR MULTI-CHANNEL MONITORING OF MEASURED QUANTITIES VIA USB

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN NEVORAL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JOSEF STRNADEL, Ph.D.

BRNO 2013

Abstrakt

Cílem této práce je navrhnout a sestrojit zařízení s mikrokontrolérem Freescale MC9S08JM60, které bude rozšiřovat laboratorní přípravek používaný ve cvičeních z předmětu *Prvky počítačů* na FIT VUT v Brně. Zařízení umožňuje sledování několika měřených veličin a odesílání naměřených hodnot do počítače přes rozhraní USB.
První část práce představuje současný laboratorní přípravek a použitý mikrokontrolér. Druhá obsahuje návrh zařízení včetně schématu, desky plošných spojů a použitých součástek, popis implementace firmwaru pro mikrokontrolér v jazyce C a řídící aplikace v jazyce C#. Ta přijímá naměřené vzorky dat ze zařízení, vizualizuje je, a tak demonstruje funkčnost zařízení. Závěrečná část navrhuje kroky pro inovaci přípravku založené na tomto zařízení, obsahuje jeho zhodnocení a možná rozšíření.

Abstract

The aim of this work is design and construction of a device based on microcontroller Freescale MC9S08JM60 that will extend laboratory kit used in the lab of Computer Hardware course at FIT BUT. The device allows monitoring of several measured quantities and sending the measured values to a computer via USB.
The first part presents the current laboratory kit and used microcontroller. The second part contains device's design including schematic, PCB and a list of used electronic components, a description of implementation of firmware for the microcontroller in C and control application written in C#. This application receives the measured data samples from the device and visualizes them to demonstrate the functionality of the device. In the last part of the bachelor's thesis there are proposed steps to upgrade the laboratory kit based on this device, contained its evaluation and possible extensions.

Klíčová slova

Freescale, MC9S08JM60, USB komunikace, USB modul, A/D převodník.

Keywords

Freescale, MC9S08JM60, USB communication, USB module, A/D converter.

Citace

Nevoral Jan: Návrh a realizace prototypu zařízení pro vícekanálové sledování měřených veličin po USB, bakalářská práce, Brno, FIT VUT v Brně, 2013

Návrh a realizace prototypu zařízení pro vícekanálové sledování měřených veličin po USB

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Josefa Strnadela, Ph.D.
Další informace mi poskytl Ing. Michal Bidlo, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jan Nevoral
14.5.2013

Poděkování

Tímto bych chtěl poděkovat Ing. Josefu Strnadelovi, Ph. D., za vedení bakalářské práce, zpřístupnění potřebného technického vybavení a čas strávený při konzultacích.

Obsah

1 Úvod

K demonstraci vyučované látky ve cvičeních z předmětu *Prvky počítačů* na FIT VUT Brno slouží laboratorní přípravek, který obsahuje mimo základních elektronických součástek a obvodů A/D převodník pro sledování signálu. Naměřené hodnoty jsou odesílané do počítače pomocí sériového přenosu.

Cílem této práce je vytvořit prototyp zařízení s mikrokontrolérem Freescale MC9S08JM60, který by umožňoval sledovat více měřených veličin, komunikoval s počítačem přes rozhraní USB a mohl být využit při inovaci přípravku místo současného jednokanálového A/D převodníku.

V první části práce popisuji současný laboratorní přípravek (kapitola 2.1) a představuji použitý mikrokontrolér (kapitola 2.2). Následující část obsahuje návrh zařízení včetně schématu, desky plošných spojů a použitých součástek (kapitola 3.1). V kapitole 3.2 popisuji implementaci firmwaru pro mikrokontrolér, za ní následuje kapitola s popisem implementace řídící aplikace, která sbírá naměřené vzorky dat, vizualizuje je, a demonstruje tak funkčnost zařízení. Konec práce navrhuje kroky pro inovaci přípravku založené na tomto zařízení (začátek kapitoly 3.4) a jeho možná rozšíření (kapitola 3.4.2). V závěru hodnotím celou práci, uvádím faktory ovlivňující přesnost naměřených hodnot a přidávám náměty na její další pokračování.

2 Použité prostředky

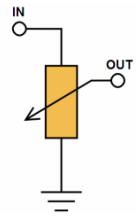
2.1 Laboratorní přípravek – současný stav

Jedním z povinných předmětů bakalářského studia na FIT VUT je předmět *Prvky počítačů (zkráceně IPR)*. K demonstraci vyučované látky v tomto předmětu slouží ve cvičeních laboratorní přípravek (zkráceně KIT, Obrázek 2.1), který obsahuje studentem libovolně propojitelné moduly – potenciometry, invertory, integrátory, diody, napájecí zdířky, sumátor a zdířku pro vstup napěťového signálu do vestavěného A/D převodníku. KIT je připojen přes rozhraní RS-232 k počítači, na kterém je možné pomocí aplikace *IPR metr* zobrazovat hodnoty z A/D převodníku. Informace v této kapitole byly čerpány z [1] a elektrického schématu zapojení KITu.

Obrázek 2.1 – Laboratorní přípravek [2]

2.1.1 Moduly

Potenciometry se využívají jako napěťové děliče. Jeden konec odporové dráhy je uzemněn, jezdec a druhý konec odporové dráhy jsou vyvedeny do zdířek na KITu (Obrázek 2.2).

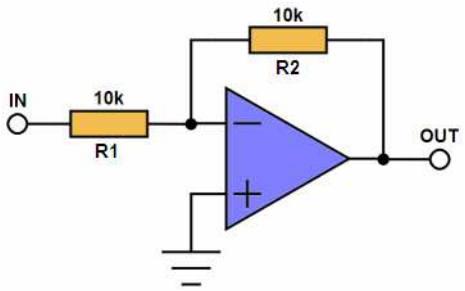


Obrázek 2.2 – Potenciometr

Invertory jsou prvky, které mění polaritu vstupního napětí. Signál je přiveden na invertující vstup operačního zesilovače (zkráceně OZ), který má pomocí rezistorů nastavené zesílení rovno mínus jedné. Neinvertující vstup tohoto zesilovače je uzemněn (Obrázek 2.3).

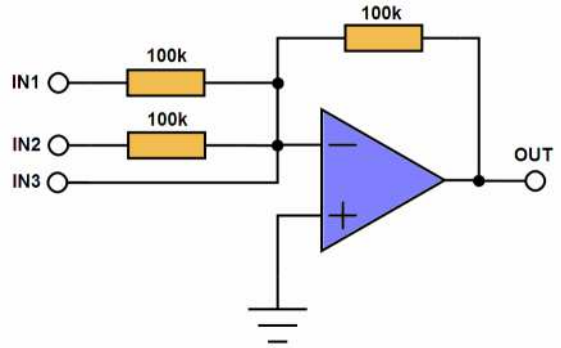
Operační zesilovač je univerzální analogový elektrický obvod, který funguje jako diferenciální napěťový zesilovač s velikým vnitřním ziskem bez zpětné vazby. Má dva vstupy – invertující (ve schématech označen znakem „-“) a neinvertující (označen „+“), které mají teoreticky nekonečný vstupní odpor. Výstupní odpor je teoreticky nulový, proto mívají OZ často jištění proti zkratu na výstupním obvodu. Operační zesilovače mají malou závislost vlastností na teplotě a napájecím napětí [3].

Na schématu invertoru (Obrázek 2.3) je jedno ze základních zapojení operačního zesilovače. Odpor mezi invertujícím vstupem a výstupem (R2) představuje jeho *zpětnou vazbu*. Zesílení vstupního signálu je dáno vztahem -R2/R1 [3].



Obrázek 2.3 – Schéma invertoru

Sumátor (také směšovač nebo sumační zesilovač) invertuje součet napětí vstupních signálů. Je realizován opět pomocí operačního zesilovače, na jehož invertující vstup jsou přes odpory připojeny dva vstupy sumátoru (IN1, IN2 – viz Obrázek 2.4). Třetí (IN3) je přiveden přímo na invertující vstup operačního zesilovače, takže je jeho napětí násobeno (teoreticky) nekonečnem. V praxi jsme omezeni jeho maximálním resp. minimálním výstupním napětí, které závisí na typu OZ a jeho napájecím napětí. Neinvertující vstup OZ je opět uzemněn.

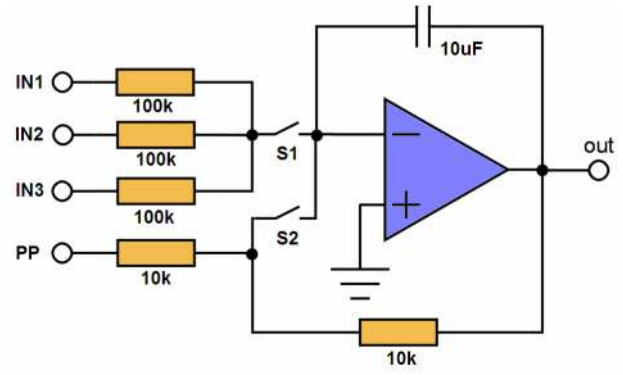


Obrázek 2.4 – Schéma sumátoru

Integrátor je realizován také pomocí operačního zesilovače, jehož neinvertující vstup je uzemněn (Obrázek 2.5). Na KITu můžeme přepínat mezi dvěma jeho stavy lišícími se sepnutím resp. rozepnutím spínačů S1 a S2.

- **Počáteční podmínka** – Spínač S1 je rozepnut a S2 sepnut. Přes vstup *PP* je možné nastavit počáteční podmínku integrátoru. Kondenzátor ve zpětné vazbě OZ se nabije napětím přivedeným na vstup *PP*.

- **Řešení** – Spínač S1 je sepnut a S2 rozepnut. Integrátor integruje napětí sečtené z jeho vstupů (IN1, IN2 a IN3). Integrace začíná od napětí, na které byl před započetím řešení nabit kondenzátor.



Obrázek 2.5 – Schéma integrátoru

Studenti modelují ve cvičeních pomocí těchto modulů diferenciální rovnice. V některých hodinách požívají z KITu pouze A/D převodník, neboť plní funkci voltmetru.

2.1.2 Napájení, mikrokontrolér, A/D převodník

Součástí KITu jsou i zdířky s **napájecím napětím**. Studenti mají k dispozici nesymetrické stabilizované napětí +5 V a symetrické stabilizované napětí ±10 V (obě jsou stabilizována ze symetrického napětí ±18 V). Tato napětí slouží také jako napájecí napětí pro vnitřní obvody KITu včetně mikrokontroléru, kterým je laboratorní přípravek také vybaven. Jedná se o osmibitový mikrokontrolér AT89C52 firmy Atmel. Má za úkol pomocí externího D/A převodníku a komparátorů plnit funkci A/D převodníku, komunikovat přes COM port (jsou přenášeny hodnoty z A/D převodníku) nebo mj. odpojovat napájecí napětí ±10 V v případě velkého odebíraného proudu – např. při zkratu.

A/D převodník umožňuje sledovat pouze jeden kanál, vzorkování má bitovou hloubku 8 bitů. V dnešní době už není KIT připojen k počítači přes rozhraní RS-232, ke každému byl nainstalován RS-232/USB převodník a pro přenos dat se využívá rozhraní USB.

Do budoucna se plánuje modernizace tohoto přípravku. Jeho součástí by mělo být zařízení, které umožňuje sledovat větší počet měřených kanálů s větší vzorkovací frekvencí a bitovou hloubkou. Ke komunikaci s počítačem by již mělo využívat přímo rozhraní USB. O návrhu takovéhoto zařízení pojednávají následující kapitoly.

2.2 Inovace na bázi MC9S08JM60

MC9S08JM60 (zkráceně JM60) je osmibitový mikrokontrolér firmy Freescale. Je náplní několika přednášek povinného předmětu *Mikroprocesorové a vestavěné systémy (zkráceně IMP)* bakalářského studia na FIT VUT Brno, proto zde nebudu detailně popisovat veškeré jeho moduly, ale vyberu pouze ty, které jsou potřeba pro vytvoření firmwaru pro JM60 v této práci. Detaily o ostatních modulech lze nalézt v katalogovém listu [4] nebo přednáškách předmětu IMP. Následující informace (pokud nebude uvedeno jinak) jsem čerpal právě z katalogového listu [4].

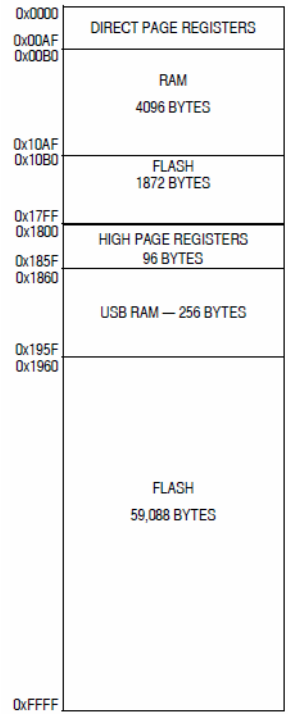
2.2.1 CPU

JM60 je mikrokontrolér s Von Neumannovou architekturou, jeho CPU je postaveno na střadačové architektuře s instrukční sadou HC08 (rozšířenou o BGND instrukce) typu CISC. Frekvence jeho HCS08 CPU může být až 48 MHz, frekvence interní sběrnice 24 MHz. Podporuje až 32 zdrojů přerušení. Šířka jeho datové sběrnice je 8 b a adresové sběrnice 16 b.

2.2.2 Paměť

Registry i paměť jsou mapovány do společného adresového prostoru, který zahrnuje 4 kB RAM, 256 B USB RAM, cca 60 kB FLASH paměti a registry (řídící/stavové, registry pro I/O).

RAM paměť je statická paměť mapovaná na adresy 0x0000 – 0x10AF a 0x1860 – 0x195F (USB RAM, Obrázek 2.6). Uchovává v sobě data po celou dobu, kdy je připojena ke zdroji elektrického napětí. Po jeho odpojení mají její buňky nedefinovanou hodnotu. K prvním 256 bytům se může přistupovat efektivněji pomocí 8b adresace.



Obrázek 2.6 – Struktura paměti JM60 [4, Figure 4-1]

FLASH paměť je primárně určena pro uložení programu řídící mikrokontrolér, má možnost zabezpečení dat. V mikrokontroléru JM60 zaujímá kapacitu 60912 B, je umístěna na adresovém rozsahu 0x10B0 – 0x17FF a 0x1960 – 0xFFFF.

Paměť pro registry se dělí na: Direct-page (0x0000 – 0x00AF), High-page (0x1800 – 0x185F) a Non-volatile (0xFFB0 – 0xFFBF), za kterou následuje tabulka vektorů přerušení (0xFFC0 – 0xFFFF).

2.2.3 Periferie JM60

Mikrokontrolér má vlastní zdroj hodinového signálu díky MCG (multi-purpose clock generator) modulu, ale je k němu možné připojit externí krystal nebo rezonátor. MCG obsahuje moduly pro PLL (phase locked loop) i FLL (frequency locked loop).

Mimo MCG obsahuje Freescale MC9S08JM60 také následující moduly:

- ACMP (analog comparator)
- ADC (analog-to-digital converter)
- COP (computer operating properly)
- I^2C (inter-integrated circuit)
- KBI (keyboard interrupt module)
- RTC (real-time counter)
- dva SCI (serial communication interface) moduly
- dva SPI (serial peripheral interface) moduly
- dva šestnáctibitové TPM (timer/pulse-width modulator) moduly
- USB (universal serial bus)

Pro řešení práce mi bude stačit jen A/D převodník (ADC), USB modul a čítač reálného času (RTC).

USB modul

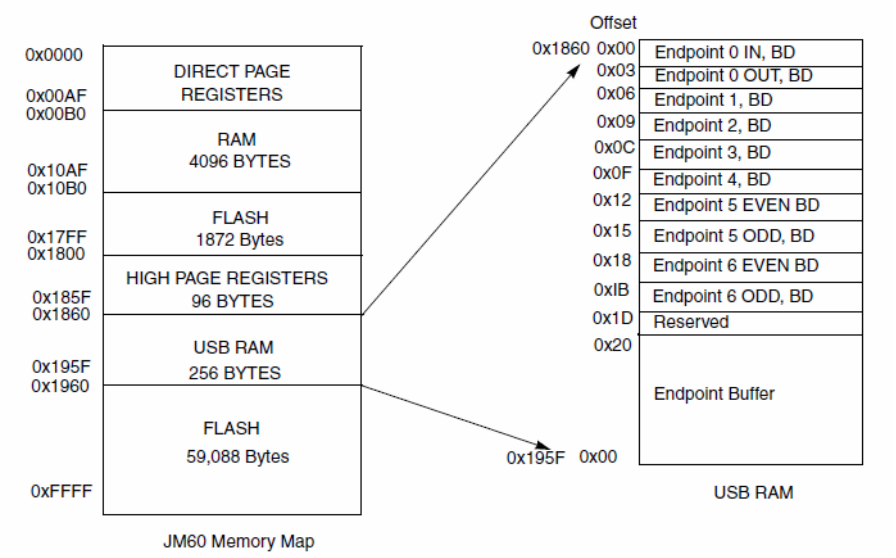
Pro rozhraní USB slouží v mikrokontroléru USB modul, který podporuje USB 2.0 full-speed (12Mbps) a všechny typy přenosů (řídící, izochronní, přerušovací i dávkové – viz kapitola 2.2.7). Pracuje nezávisle na činnosti mikroprocesoru, je ovládán řídícími a stavovými registry a může vyvolávat přerušení mikroprocesoru. Modul obsahuje svůj 3,3V napěťový regulátor a USBDP pull-up rezistor. USB zařízení s tímto mikrokontrolérem může využít až 7 endpointů (viz kapitola 2.2.7). Nultý je vždy povinný, dalších šest je volitelných. Nultý a poslední dva jsou obousměrné.

Paměťový prostor obsahuje 256 B USB RAM paměti mapované do adresového prostoru 0x1860 - 0x195F (Obrázek 2.7). Na jejím počátku je BDT (buffer descriptor table), což je tabulka tzv. „buffer deskriptorů“ – registrů pro řízení EP. Od adresy 0x1880 zbývá prostor, který může být vyžit pro data-buffery endpointů (Obrázek 2.8). Tam se ukládají data přijatá od počítače, nebo vysílaná počítači.

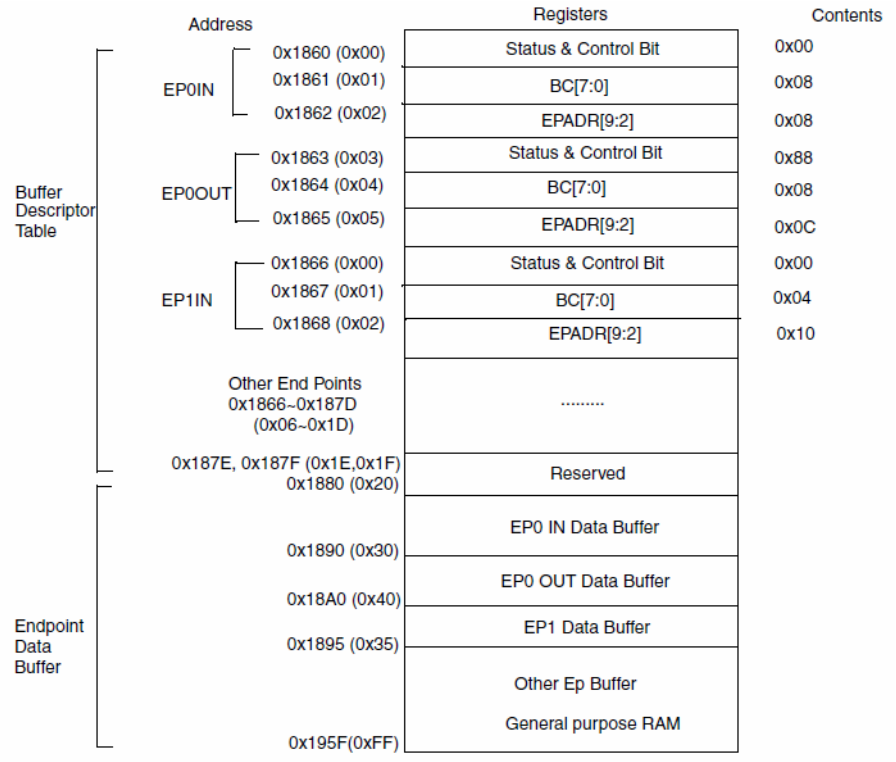
K mikrokontroléru MC9S08JM60 jsou volně k dispozici USB ovladače, takže můžeme implementovat komunikaci přes USB na vyšší úrovni abstrakce. Mikrokontrolér podporuje i Freescale USB-LITE stack od CMX, který zajišťuje podporu tříd HID a CDC.

A/D převodník

Dvanáctibitový A/D převodník mikrokontroléru JM60 v 64pinovém pouzdru umožňuje využít až dvanáct jeho pinů jako své kanály, ostatní typy JM60 pouze 8 pinů. V registrech (APCTL1, APCTL2 a APCTL3) se kanály povolují jako vstupy převodníku. Ten pracuje nezávisle na činnosti mikroprocesoru, je ovládán řídícími a stavovými registry (ADCSC1, ADCSC2) a může vyvolávat přerušení mikroprocesoru.



Obrázek 2.7 – USB RAM [5, Figure 3]



Obrázek 2.8 – Příklad konfigurace BDT [5, Figure 7]

A/D převodník umožňuje 12ti, 10ti a 8mi bitový převod (nastavuje se v registru ADCCFG), výsledek je indikován bitem COCO v ADCSC1 a číselná hodnota je potom uložena v registrech ADCRH a ADCRL. Je možné také nastavit, že bude indikován výsledek A/D převodu pouze v případě, že je menší (popř. větší nebo roven) než hodnota uložená v registrech ADCCVH a ADCCVL.

V konfiguračním registru (ADCCFG) lze kromě bitové hloubky převodu volit i zdroj hodinového signálu, jeho dělící poměr z daných možností a povolit nízkopříkonový režim nebo delší dobu vzorkování.

RTC modul

Protože může být potřeba v mikrokontroléru provádět některé funkce periodicky, má JM60 k dispozici čítač reálného času (RTC). Jeho výstupem je periodický signál s uživatelem danou periodou z tabulky (Tabulka 2.1). Časové intervaly v této tabulce mají jednotky z reálného světa – podle názvu modulu.

Modul pracuje nezávisle na činnosti mikroprocesoru, je ovládán řídícím a stavovým registrem RTCSC, který určuje zdroj hodinového signálu (RTCLKS), jeho dělící poměr (RTCPS) a povoluje přerušení mikroprocesoru. To vzniká při samotném čítání, které probíhá v registru RTCCNT (je pouze pro čtení) od nuly do hodnoty dané registrem RTCMOD právě když by hodnota v RTCCNT měla být vyšší než v RTCMOD a začíná se tedy s čítáním opět od nuly.

RTCPS	1-kHz Internal Clock (RTCLKS = 00)	1-MHz External Clock (RTCLKS = 01)	32-kHz Internal Clock (RTCLKS = 10)	32-kHz Internal Clock (RTCLKS = 11)
0000	Off	Off	Off	Off
0001	8 ms	1,024 ms	250 µs	32 ms
0010	32 ms	2,048 ms	1 ms	64 ms
0011	64 ms	4,096 ms	2 ms	128 ms
0100	128 ms	8,192 ms	4 ms	256 ms
0101	256 ms	16,4 ms	8 ms	512 ms
0110	512 ms	32,8 ms	16 ms	1,024 s
0111	1,024 s	65,5 ms	32 ms	2,048 s
1000	1 ms	1 ms	31,25 µs	31,25 ms
1001	2 ms	2 ms	62,5 µs	62,5 ms
1010	4 ms	5 ms	125 µs	156,25 ms
1011	10 ms	10 ms	312,5 µs	312,5 ms
1100	16 ms	20 ms	0,5 ms	0,625 s
1101	0,1 s	50 ms	3,125 ms	1,5625 s
1110	0,5 s	0,1 s	15,625 ms	3,125 s
1111	1 s	0,2 s	31,25 ms	6,25 s

Tabulka 2.1 – Perioda hodin pro RTC [4, Table 13-6]

2.2.4 Vstupně výstupní porty

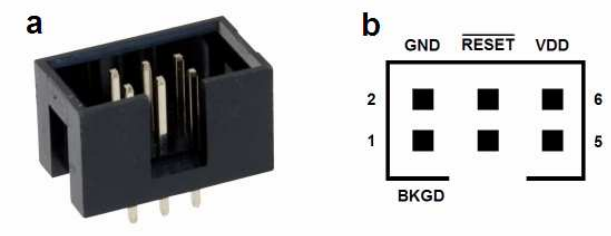
JM60 má až 51 univerzálních vstupně/výstupních pinů (GPIO – general purpose input/output) rozdělených do sedmi portů (port A – port G). Pomocí registrů se může každý pin nastavit jako vstupní nebo výstupní, ke každému vstupnímu připojit interní pull-up rezistor. Mikrokontrolér se vyrábí ve více provedeních, které se liší použitými pouzdry (tudíž rozměry), počty vývodů (a tedy i počty vstupně/výstupních portů a pinů), ale např. i počty kanálů A/D převodníku:

- MC9S08JM60CGT – 48 pinů, pouzdro QFN48, rozměry: 7 mm x 7 mm x 1,0 mm
- MC9S08JM60CLD – 44 pinů, pouzdro LQFP44, rozměry: 10 mm x 10 mm x 1,4 mm
- MC9S08JM60CLH – 64 pinů, pouzdro LQFP64, rozměry: 10 mm x 10 mm x 1,4 mm
- MC9S08JM60CQH – 64 pinů, pouzdro QFP64, rozměry 14 mm x 14 mm x 2,2 mm

2.2.5 Programování a ladění

Výrobce mikrokontrolérů MC9S08 firma Freescale nabízí kromě mikrokontrolérů také své vývojové prostředí CodeWarrior. To lze využít i při programování JM60. Firmware lze vytvářet buď čistě v assembleru (instrukce: [4], Table 7-2) nebo např. v jazyce C. Z něho se potom kód do instrukcí pro mikroprocesor překládá.

Programování mikrokontroléru probíhá přes pin BKGD/MS. Ten je obvykle vyveden do konektoru BDM (Obrázek 2.9a) na desce s mikrokontrolérem. Do tohoto konektoru je kromě BKGD (pin 1) přivedeno také napájecí napětí mikrokontroléru (pin 6), zem (pin 2) a vývod pro jeho resetování (pin 4). Zbylé piny jsou neobsazené (Obrázek 2.9b).

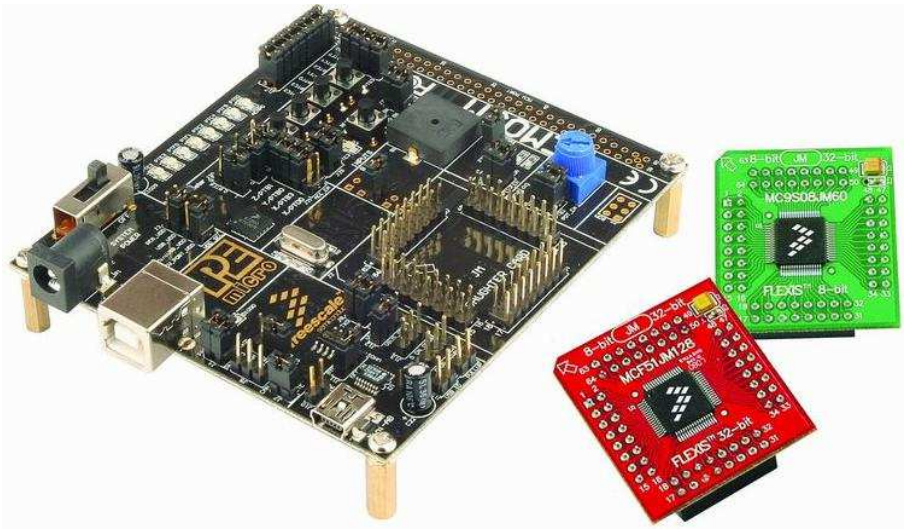


Obrázek 2.9 – Konektor BDM: a) vzhled [6] b) zapojení pinů

BDM (background debug mode) je ladící rozhraní mikrokontrolérů Freescale. Slouží k nahrání programu do mikrokontroléru, dovoluje sledovat jeho činnost a ladit i při běhu programu. Pokud je po resetu aktivní pin BKGD, dostává se mikrokontrolér do režimu active background mode, v němž je možné mu zasílat příkazy např. na zjištění/nastavení obsahu registrů nebo krokování programu.

2.2.6 DEMOJM

Pro vývoj jednoduchých zařízení založených na JM60 může stačit *DEMOJM* (Obrázek 2.10). Je to levný vývojový přípravek pro mikrokontroléry MC9S08JM60 a MCF51JM128 firmy Freescale. Součástí tohoto přípravku jsou mj. USB konektor, tříosý akcelerometr, piezo bzučák, trimr, 8 LED diod a 4 tlačítka. Na desce je zabudován také obvod *P&E Embedded Multilink*, který umožňuje spojit desku DEMOJM přes USB přímo s počítačem a slouží jako BDM k programování mikrokontroléru a ladění.



Obrázek 2.10 – DEMOJM [7]

2.2.7 Návrh USB zařízení s JM60

Při návrhu USB zařízení založeném na JM60 je nutné vytvořit:

- **Hardware** – zařízení vybavené mikrokontrolérem MC9S08JM60 . V této práci se jím zabývá kapitola 3.1.
- **Firmware** – program, který bude řídit mikrokontrolér (a jeho vestavěný USB modul). Bude popsán v kapitole 3.2.
- **Software** – ovladač zařízení, případně i obslužný program pro hostitelské zařízení (kapitola 3.3)

Princip rozhraní USB [8]

USB je asynchronní sériové rozhraní postavené na architektuře master-slaves (host-devices). **Host** (většinou počítač) řídí veškerou komunikaci s připojenými **zařízeními** (angl. device), řeší jejich připojování, odpojování a počáteční konfiguraci (tzv. enumerace).

K hostovi lze pomocí rozbočovačů (angl. USB hubs) připojit až 127 zařízení, každé má svoji adresu v rozmezí 1 až 127, kterou mu přidělí při enumeraci. Na adresu 0 odpovídá nově připojené zařízení, kterému ještě nebyla adresa přidělena.

Přenos informací přes rozhraní USB probíhá na tzv. „endpointech“ (zkráceně EP) – adresovatelných místech sloužících pro vysílání/příjem dat. Endpointy mohou být dvojího typu:

- **Příchozí EP** (endpoint OUT): přenos z hosta do zařízení
- **Odchozí EP** (endpoint IN): přenos ze zařízení do hosta

USB zařízení má většinou více endpointů (JM60 jich má 7), každé musí implementovat obousměrný EP0 pro enumeraci, řídicí a stavové přenosy.

Ovladač v počítači potom komunikuje přes tzv. „roury“ (angl. USB pipes), což je spojení s některým z endpointů zařízení. Každé zařízení má vytvořenou rouru mezi počítačem a endpointem 0.

Režimy rychlosti USB 2.0

USB modul v mikrokontroléru JM60 podporuje režim USB 2.0 Full-Speed (viz kapitola 2.2.3). Je to jeden ze tří rychlostních režimů, které se liší např. maximální velikostí paketů u různých typů přenosů (viz dále) [8]:

- **Low-speed** – rychlost až 1,5 Mbps
- **Full-speed** – rychlost až 12 Mbps
- **High-speed** – rychlost až 480 Mbps

Typy přenosů

Rozlišujeme 4 typy přenosů na sběrnici USB. Každý používaný endpoint zařízení provádí přenos některého z následujících typů [8], [9], [10]:

- **Řídicí (control transfer)** – obousměrný počítačem řízený přenos příkazů, stavových informací apod. Je využíván na endpointech 0 IN a 0 OUT např. v procesu enumerace (viz

dále). Maximální velikost paketu je 8 B pro USB režim Low-speed, 8 B, 16 B, 32 B, 64 B pro Full-speed a 64 B pro High-speed.

- **Přerušení (interrupt transfer)** – přenos s garantovanou latencí obsluhy. Používá se především pro přenos stavových informací, znaků nebo dat v definovaných intervalech činnosti zařízení. Velikost paketu může být až 8 bytů v režimu Low-speed, 64 bytů v režimu Full-speed a až 1024 bytů v režimu High-speed.
- **Dávkový (bulk transfer)** – Využívá se především pro přenosy velkého množství dat. Pakety mohou mít velikost 8, 16, 32 nebo 64 bytů v režimu Full-speed až 512 bytů v režimu High-speed. Je vybaven protokolem pro detekci a opravu chyb, a proto garantuje bezchybný přenos. Nemá garantovanou šířku pásma, proto se dávkové přenosy realizují tehdy, když není potřeba vykonávat přenosy jiného typu.
- **Isochronní (isochronous transfer)** – nejčastěji používán při proudových přenosech (audio, video). Má garantovanou šířku pásma, ale nezaručuje bezchybné přenesení USB paketu. Ty mohou mít velikost až 1023 B pro USB režim Full-speed (1024 B pro High-speed). Minimální interval, ve kterém je možné posílat pakety, je 1 ms pro režim Full-speed a 125 mikrosekund pro režim High-speed [11].

Identifikátory zařízení [12]

Každý typ zařízení, které se připojuje k USB, by měl mít svoji jednoznačnou identifikaci. K tomuto účelu slouží dvě 16-bitová identifikační čísla: **Vendor ID** (zkráceně VID) a **Product ID** (zkráceně PID). VID je identifikační číslo výrobce, které mu přidělila organizace USB-IF. PID slouží pro označení jednotlivých typů výrobků (ne jednotlivých kusů) a přiřazuje ho už výrobce sám.

Bez unikátní kombinace těchto identifikátorů se lze obejít, pokud je zařízení pouze experimentální nebo je nasazeno v minimálním počtu kusů v místech, kde nemůže dojít ke kolizi dvou stejných VID a PID.

Enumerace [8]

Zařízení musí po připojení k PC reagovat na jeho výzvy (ty jsou dány standardem USB) a podle nich zasílat počítači požadované informace. Kvůli tomu musí mít každé zařízení implementovaný obousměrný endpoint 0, který realizuje řídící přenosy (control transfers). Celý proces počátečního vyjednávání se nazývá enumerace. Je řízena počítačem, na jejím konci je v počítači zaveden ovladač a zařízení připraveno k funkci. Skládá se z:

- Resetu zařízení
- Žádosti o zaslání deskriptoru zařízení na endpoint 0
- Přidělení adresy zařízení (po připojení musí být nastavena na nulu)
- Zaslání dalších deskriptorů

Deskriptory obsahují VID, PID, informace o konkrétním zařízení, jeho konfiguraci, endpointech, atd. Obsah deskriptorů a jejich výčet je možné nalézt v [13].

3 Návrh a řešení

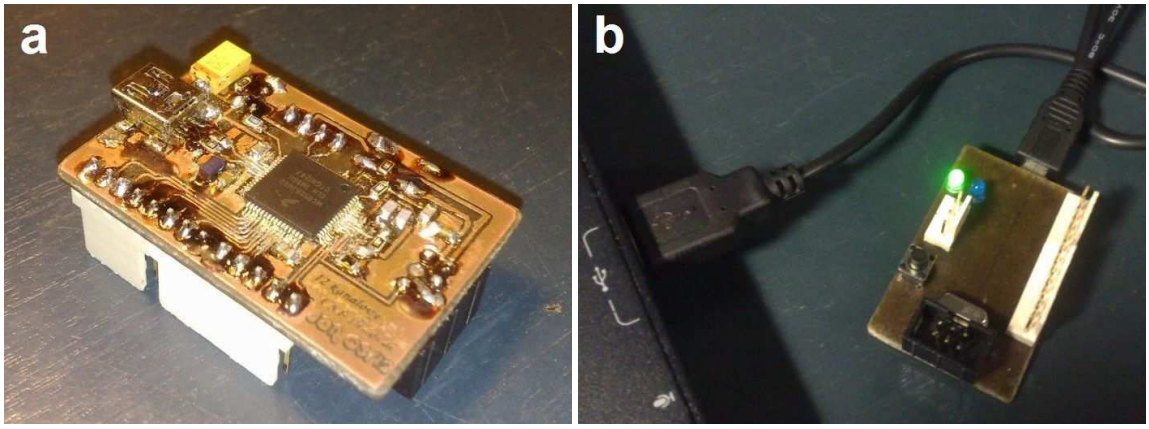
Mým prvním úkolem bylo vytvořit jednoduché USB zařízení s pomocí mikrokontroléru MC9S08JM60, které bude zajišťovat pravidelný přísun dat pro připojený počítač. Jelikož je to jenom elementární část celého zařízení, které má vzorkovat několik souběžných veličin z jeho okolí, budu popisovat rovnou celé zařízení.

Zadání mi nespecifikovalo příliš mnoho konkrétních požadavků na výsledné zařízení, proto jsem se ho rozhodl navrhovat tak, aby šlo využít ve cvičeních předmětu IPR jako část laboratorního přípravku, aby zde splňovalo potřeby studentů i učitelů a aby byly co nejvíce využity možnosti, které JM60 nabízí – ty jsou také v mnoha ohledech limitující (např. velikost USB RAM, frekvence CPU).

Při vývoji firmwaru jsem použil vývojový kit *DEMOJM* (viz kapitola 2.2.6), díky němuž jsem se nejprve s mikrokontrolérem seznamoval. Po zprovoznění jeho komunikace s počítačovou aplikací přes rozhraní USB jsem navrhnul a vyrobil prototyp zařízení a dále vyvíjel firmware přímo pro něj.

3.1 Prototyp zařízení

Prototyp zařízení představuje deska plošných spojů osazená mikrokontrolérem JM60 v 64pinovém pouzdru a dalšími elektrickými součástkami důležitými pro funkci zařízení (Obrázek 3.1).

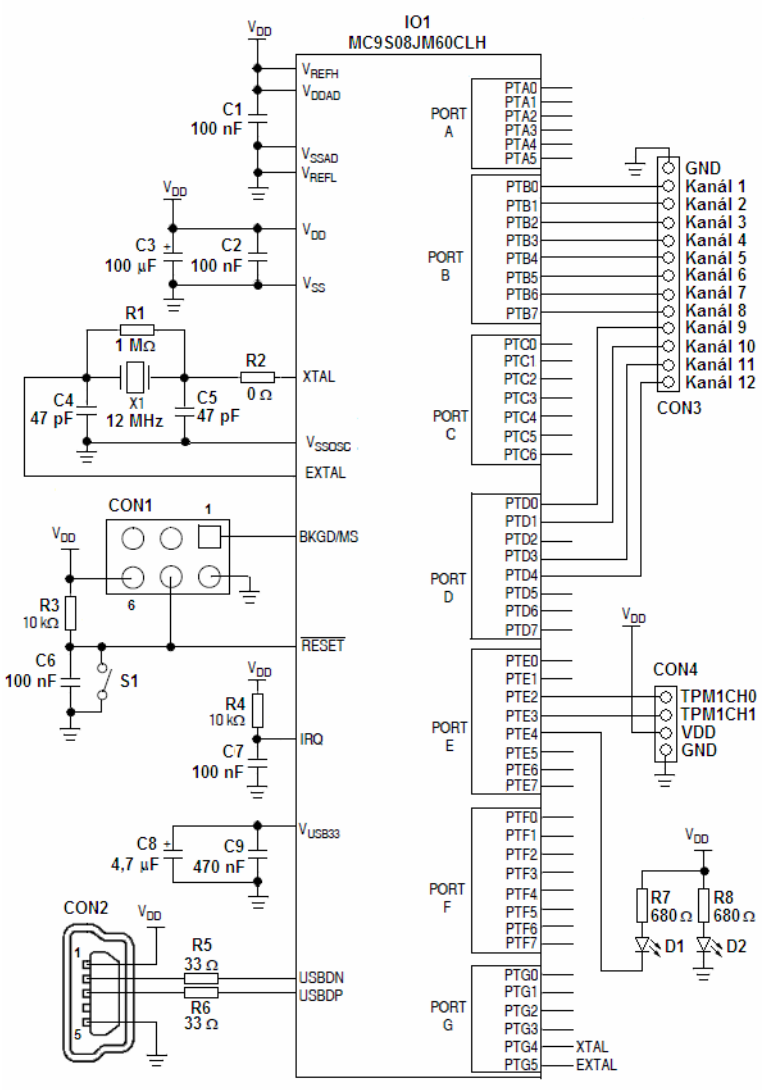


Obrázek 3.1 - Prototyp zařízení a) pohled ze strany spojů b) pohled ze strany součástek

3.1.1 Schéma zapojení

Při návrhu zařízení jsem vycházel ze základního zapojení JM60 [4, Figure 2-4]. Snažil jsem se vytvořit co nejjednodušší zapojení, proto je zařízení napájeno z počítače, který poskytuje stabilizované stejnosměrné napětí 5V, přes USB kabel, a mikrokontrolér využívá svůj 3,3V regulátor pro USB modul. Schéma zapojení prototypu zařízení viz Obrázek 3.2.

Napájecí napětí mikrokontroléru je filtrováno kondenzátorem C2 (keramický) a stabilizováno kondenzátorem C3 (tantalový), napájecí napětí A/D převodníku je filtrováno (od vysokofrekvenčního šumu) kondenzátorem C1 (keramický). Stejně tak musí být stabilizováno a filtrováno napětí vnitřního zdroje 3,3 V pro USB modul. K tomu slouží kondenzátory C8 a C9.



Obrázek 3.2 – Schéma zařízení

Zařízení využívá USB modul, takže je nutné připojit obvod s externím krystalem [4, Figure 2-4, NOTE 1]. Ten obsahuje kromě samotného krystalu (X1) odpory R1 a R2 a kondenzátory C4 a C5. Odpor R1 je volen z rozmezí 1 – 10 MΩ a jeho hodnota není obvykle rozhodující, rezistor R2 v desítkách ohmů (i 0Ω). Kondenzátory C1 a C2 jsou většinou zvoleny stejné, jejich kapacita se počítá jako dvojnásobek zatěžovací kapacity krystalu (závisí na použitém krystalu), od které se odečte kapacita mezi piny mikrokontroléru a kapacita mezi vodivými spoji na DPS (cca 10 pF). Měly by být použity kvalitní keramické kondenzátory a odpory s malou indukcí (např. uhlíkové). [4, kapitola 2.3.2].

K programování mikrokontroléru slouží BDM konektor ve schématu označený jako CON1, k manuálnímu resetu tlačítko S1. Do cesty signálových vodičů z USB konektoru (CON2) do mikrokontroléru jsou oproti zapojení, ze kterého jsem vycházel, vloženy odpory R5 a R6 (oba 33Ω ± 1%) pro dodržení USB specifikace [4, kapitola 17.2.1 a 17.2.2].

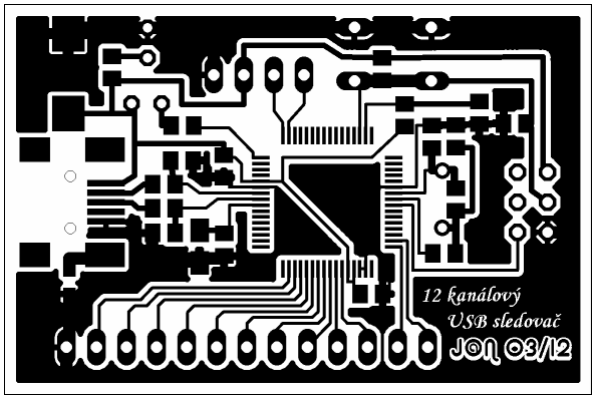
Zařízení má dvě LED diody: D2 svítí, pokud je zařízení napájeno. D1, připojena na pin 4 portu E, slouží k indikaci spuštěného vzorkování kanálů a její svit ovládá mikrokontrolér. Piny 2 a 3 téhož portu (napojeny na dva kanály TPM modulu JM60) jsou přivedeny na konektor CON4, který zatím nemá využití, zamýšlel jsem ho pro výhledová rozšíření zařízení. Je možné je využít jako obecné vstupně/výstupní piny mikrokontroléru nebo pro generování obdélníkových signálů.

Na konektor CON3, připravený pro sledované signály, je kromě dvanácti kanálů A/D převodníku vyvedena zem (nulový potenciál), vůči které měří A/D převodník vzorkované napětí. V případě většího nebo stejného napětí jako na pinu V_{REFH} (tedy 5 V) je výsledkem číselného převodu maximální hodnota.

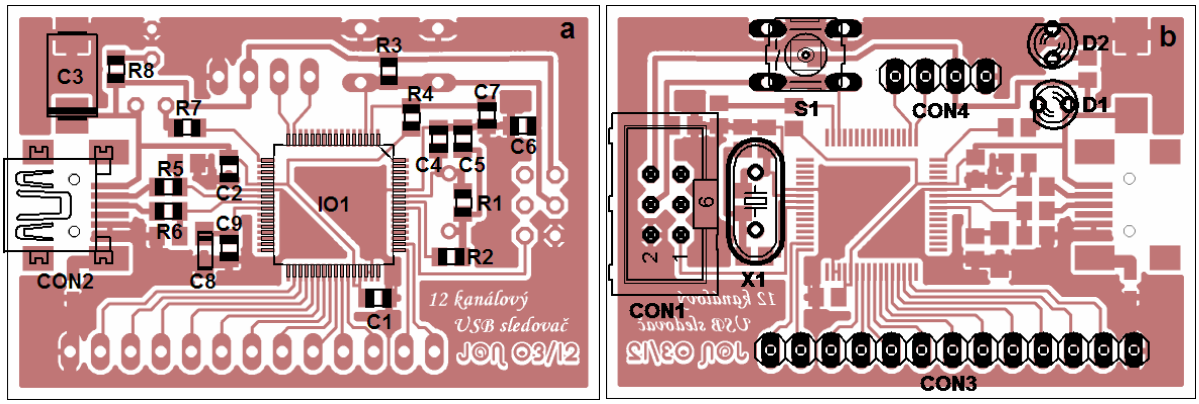
3.1.2 Deska plošných spojů

Desku plošných spojů (DPS) jsem navrhoval v programu *Eagle 6.4.0*, který se pro takovéto účely běžně používá. Pro vytvoření DPS tohoto zařízení dostačuje bohatě jeho neplacená verze.

Podle schématu zapojení (Obrázek 3.2) jsem navrhnul jednostrannou desku plošných spojů o rozměrech 50 mm x 33 mm (Obrázek 3.3), respektoval jsem nutnost umístit filtrační kondenzátory co nejblíže mikrokontroléru ([4, kapitola 2.3.1]). Strana plošných spojů je osazená SMD součástkami (Obrázek 3.4a), na druhé straně je krystal a součástky, s kterými může uživatel manipulovat a které by měl vidět (Obrázek 3.4b).



Obrázek 3.3 – DPS prototypu zařízení



Obrázek 3.4 – Rozmístění součástek na DPS a) ze strany spojů b) z nepájivé strany

3.1.3 Použité součástky

Seznam použitých součástek viz Tabulka 3.1. Hodnoty součástek byly voleny podle doporučení z kapitoly 2.3 katalogového listu JM60 [4]. S krystalem zakoupeným v obchodě *GES ELECTRONICS* (dodavatel uvádí zatěžovací kapacitu 30 pF) se osvědčily kondenzátory C4 a C5 o velikosti 47 pF, odpory R1 0 Ω a R2 1 MΩ.

Odpory R5 a R6 byly vybrány tak, aby LED diodami tekl proud pod 5 mA; jejich velikost je však orientační stejně jako kapacita stabilizačního kondenzátoru C3, která by měla být minimálně v desítkách µF.

Protože se běžně neprodávají konektory s 13 piny, může se CON3 nahradit např. dvěma konektory o pěti a osmi pinech.

Součástka	Pouzdro	Počet	Označení ve schématu
Odpor 0 Ω	SMD 0805	1	R2
Odpor 33 Ω ± 1%	SMD 0805	2	R5, R6
Odpor 680 Ω ± 5%	SMD 0805	2	R7, R8
Odpor 10 kΩ ± 5%	SMD 0805	2	R3, R4
Odpor 1 MΩ ± 5%	SMD 0805	1	R1
Keramický kondenzátor 47 pF	SMD 0805	2	C4, C5
Keramický kondenzátor 100 nF	SMD 0805	4	C1, C2, C6, C7
Keramický kondenzátor 470 nF	SMD 0805	1	C9
Tantalový kondenzátor 4,7 µF	SMD A	1	C8
Tantalový kondenzátor 100 µF	SMD D	1	C3
LED dioda modrá	3 mm	1	D1
LED dioda zelená	3 mm	1	D2
Krystal 12 MHz (zatěžovací kapacita: 30 pF)	HC49/U3H	1	X1
Mikrospínač do DPS 1-pólový		1	S1
Freescale MC9S08JM60CLH	LQFP64	1	IO1
Konektor pro ploché kabely do DPS rovný, rozteč 2,54mm, 2x3 kontaktů.		1	CON1
Konektor USB Mini B, zásuvka přímá	SMD	1	CON2
Konektor se zámkem do DPS, 13 kontaktů, rozteč 2,54mm		1	CON3
Konektor se zámkem do DPS, 4 kontakty, rozteč 2,54mm		1	CON4

Tabulka 3.1 - Seznam použitých součástek

3.2 Firmware pro JM60

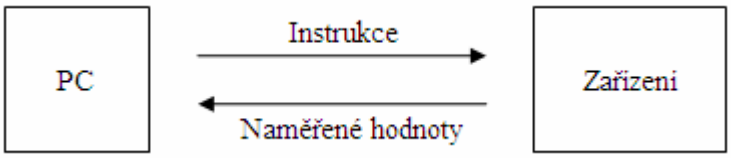
Z výše popsaných důvodů jsem se rozhodl vytvořit firmware, který půjde jednoduše modifikovat, aby mohl být mikrokontrolér (a tedy celé zařízení) v budoucnu využit pro více úloh než jen sledovaní signálů (např. výstup pulzně šířkového modulátoru, přepínání stavu integrátorů, atd.). Půjde o zařízení, které bude přijímat přes rozhraní USB od počítače předem známé instrukce a bude na ně reagovat změnou své činnosti.

O samotné vícekanálové sledování se stará A/D převodník mikrokontroléru JM60, který vzorkuje v pravidelných intervalech (ideálně co nejmenších) napětí na svých kanálech. Pokud budeme chtít sledovat jiné veličiny než napětí, anebo napětí v jiném intervalu než umožňuje A/D převodník (0 V – 5 V), budeme je nejprve muset převést na napětí v tomto měřitelném intervalu (kapitola 3.4).

JM60 (v 64pinovém pouzdru) umožňuje využít až dvanáct svých pinů jako kanály A/D převodníku (viz kapitola 2.2.3). Pokud bychom chtěli vzorkovat více kanálů, museli bychom k mikrokontroléru připojit externí multiplexor (y), kterým (i) bychom při každém vzorkování vybrali postupně všechny kanály, nebo vymyslet jiný způsob vzorkování. Pro účely laboratorních cvičení předmětu IPR však 12 kanálů bohatě dostačuje.

3.2.1 Použité typy USB přenosů

V takto navrženém zařízení potřebuji přenášet přes USB příkazy z počítače do zařízení a naměřené hodnoty z kanálů opačným směrem (Obrázek 3.5). Stačí mi tedy implementovat povinný obousměrný nultý endpoint (viz kapitola 2.2.7), jednosměrný endpoint OUT pro příkazy (přiřadím mu číslo jedna) a jednosměrný endpoint IN pro data do počítače (přiřadím mu číslo dva).



Obrázek 3.5 - Schéma komunikace

Příkazy z počítače budou posílány v nepravidelných intervalech (když bude počítačový uživatel chtít nějakou změnu) a je důležitý jejich bezchybný a včasný přenos. Zvolil jsem tedy přenos typu přerušení, neboť má oproti dávkovému vyhrazené přenosové pásmo. Isochronní přenos je svou povahou zcela nevhodný.

Naměřené hodnoty se budou odesílat v pravidelných intervalech, mělo by být zaručeno jejich bezchybné přenesení s co nejkratší časovou prodlevou kvůli jejich vykreslení nebo dalšímu zpracování. Pro jejich přenos jsem vybral také přenos přerušovaný. Protože má však stanovený maximální objem přenesených dat (je dán maximální velikost interrupt paketu – pro JM60 64 B a jejich maximální počet za sekundu – 1000), který by nemusel dostačovat, bylo by možné doplnit zařízení o další endpoint(y) – např. s dávkovým přenosem, případně použít pouze tento přenos. Jelikož se mi dařilo odesílat ze zařízení a přijímat hromadným přenosem pomocí knihovny SimpleUSB.dll (knihovna viz kapitola 3.3.2) pouze cca 64 paketů za vteřinu, volil jsem pouze přerušovaný, který nakonec k přenesení dat dostačoval.

3.2.2 Struktura paketů

Struktura a velikost paketů také nijak ze zadání práce nevyplývaly a zůstaly čistě v mojí kompetenci.

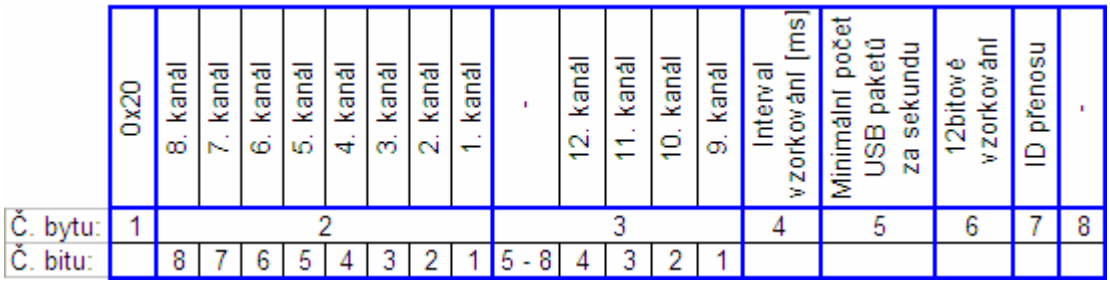
Předpokládám, že uživatel nebude chtít po celou dobu sledovat všechny kanály, ale jen některé z nabízených dvanácti a že by chtěl mít možnost kanály během sledování také měnit. Tyto změny jsou zaslány pomocí příkazu do zařízení, které změní kanály, jež vzorkuje, a začne zasílat datové pakety s nově nastavenými kanály. Protože však mohl být v této přechodné době doručen do počítače nějaký paket s předchozím nastavením kanálů, bylo potřeba od sebe tyto datové proudy nějak odlišit. Zasílám tedy s instrukcemi s žádostí o sledování kanálů (či jejich změnu) společně i identifikátor datového toku (dále jako „ID přenosu“), který je očekáván i v paketech přicházejících do počítače.

Mezi další informace v těchto paketech jsem zvolil bitovou hloubku vzorkování (A/D převodník JM60 umožňuje vzorkování s bitovou hloubkou 8, 10 nebo 12 bitů [4], naimplementovat vzorkování 8mi a 12ti bity.), interval vzorkování kanálů a minimální počet přenesených USB paketů se vzorky za sekundu. Budu-li totiž chtít se vzorky nějak dále pracovat – např. je vizualizovat, nelze čekat, než se naplní každý datový paket, ale je potřeba je do počítače odesílat častěji.

Paket s instrukcemi pro zařízení

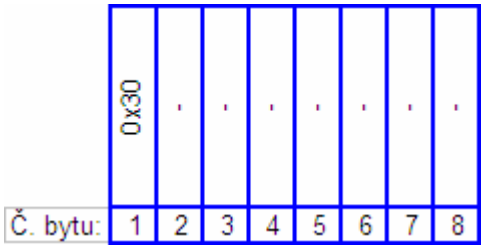
Velikost paketu s instrukcemi pro zařízení je 8 B, což poskytuje rezervu, pokud by byly v budoucnu implementovány velikostně náročnější instrukce. Do prvního bytu jsem umístil číselný kód instrukce a ostatní byty jsem ponechal pro případné doplňující informace k instrukci. Ve firmwaru mikrokontroléru i počítačové aplikaci (viz kapitola 3.3) jsou zatím implementovány tři instrukce, které pro sledování kanálů stačí. V závorce je číselný kód instrukce jako číslo v šestnáctkové soustavě:

- **Začni vzorkovat** (0x20) – Instrukce pro zahájení vzorkování kanálů. V paketu je třeba zaslat zařízení všechny již zmíněné informace. Pro přenesení informace, které kanály se mají vzorkovat, je nutné alespoň pole o 12ti bitech (12 kanálů). Každý bit v tom případě udává, zdali se má daný kanál vzorkovat, nebo ne. Pro prvních osm kanálů jsem využil všech osm bitů z druhého bytu paketu, pro zbylou čtveřici kanálů čtyři nejméně významné bity ve třetím bytu paketu. Mapování jednotlivých kanálů na bity zobrazuje Obrázek 3.6, nejméně významný bit v bytu je na něm zobrazován jako první bit. Logická jednička uložená na daném bitu znamená, že se tento kanál bude vzorkovat. Interval vzorkování v milisekundách je uložen na čtvrtém bytu, může nabývat hodnot 1 až 255. Stejných hodnot může nabývat 5. byte paketu - minimální počet USB paketů za sekundu. Nula pro tento byte není zakázaná, ale znamená, že není tento počet určen. Pro bitovou hloubku vzorkování je vyhrazen 6. byte. Jeho nenulová hodnota určuje 12b vzorkování, nulová osmibitové. ID přenosu je možné uložit na předposlední byte (může tedy nabývat hodnot 0 až 255). Zbylé části paketu, na obrázku označeny pomlčkami, neovlivňují chování zařízení.



Obrázek 3.6 – Struktura paketu instrukce *Začni vzorkovat*

- **Ukonči vzorkování** (0x30) – Instrukce pro ukončení vzorkování kanálů. Nejsou potřeba žádné doplňující informace k instrukci, takže jsou zbylé byty nevyužité (Obrázek 3.7).



Obrázek 3.7 – Struktura paketu instrukce *Ukonči vzorkování*

- **Změň vzorkované kanály** (0x40) – Instrukce pro změnu kanálů, které se mají vzorkovat. Od instrukce *Začni vzorkovat* se paket liší jen minimálně (Obrázek 3.8) – není brán zřetel na

obsah 4. a 6. bytu paketu, neboť interval a bitová hloubka vzorkování zůstávají od předešlého vzorkování zachovány. Instrukci by bylo možno nahradit sledem instrukcí *Ukonči vzorkování* a *Začni vzorkovat*.

	0x40	8. kanál	7. kanál	6. kanál	5. kanál	4. kanál	3. kanál	2. kanál	1. kanál	-	12. kanál	11. kanál	10. kanál	9. kanál	-	Minimální počet USB paketů za sekundu	-	ID přenosu	-
Č. bytu:	1	2								3					4	5	6	7	8
Č. bitu:		8	7	6	5	4	3	2	1	5 - 8	4	3	2	1					

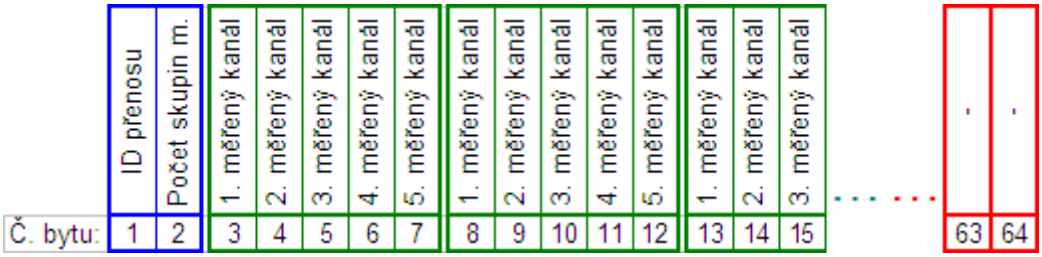
Obrázek 3.8 – Struktura paketu instrukce *Změň vzorkované kanály*

Datový paket

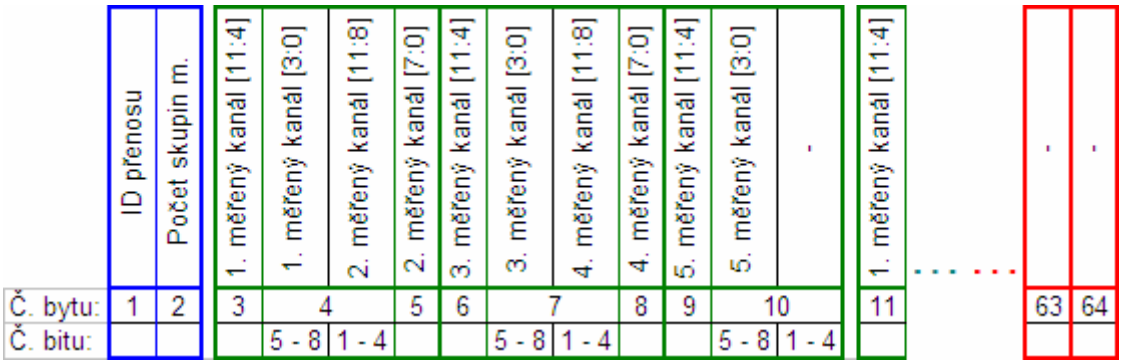
Naměřené hodnoty vzorkovaných kanálů z A/D převodníku jsou do počítače odesílány pakety přenosem typu interrupt (viz kapitola 3.2.1). Maximální velikost těchto paketů při použití USB modulu mikrokontroléru JM60 je 64 B (viz kapitola 2.2.7). Pro jednoduchost jsou všechny datové pakety stejně velké – 64 bytů, neboť chci umožnit uživatelům vzorkování s co největší frekvencí, s kterou souvisí i větší množství dat nutných přenést přes interrupt přenos rozhraní USB.

Pro další text si označme skupinu navzorkovaných hodnot sledovaných kanálů v jednom časovém okamžiku (tedy od každého sledovaného kanálu jednu hodnotu) jako *skupinu měření*. Pakety vždy obsahují nekomprimované hodnoty několika skupin měření. Jejich počet omezený velikostí paketu je určen mikrokontrolérem podle minimálního počtu přenesených paketů za sekundu z poslední instrukce. Kromě naměřených hodnot bylo tedy nutné do každého paketu uložit kromě ID přenosu i počet skupin měření (počet odesílaných naměřených hodnot dělený počtem měřených kanálů). K tomu jsem vyhradil první dva byty paketu – první určuje ID přenosu, druhý počet skupin odesílaných měření.

Strukturu paketů zobrazují obrázky: Obrázek 3.9 a Obrázek 3.10. Za prvními dvěma byty (modrou barvou) je prostor pro naměřené hodnoty. Ty jsou umístěny po skupinách měření za sebou, zbytek paketu (červenou barvou) nabývá nedefinovaných hodnot. Skupiny měření (označeny zelenou barvou) jsou vždy zarovnány na celý počet bytů. V každé skupině měření jsou za sebou umístěny hodnoty podle čísla jejich kanálů vzestupně. To znamená, že „1. měřený kanál“ na obrázcích znázorňuje sledovaný kanál s nejmenším pořadovým číslem. Při osmibitovém vzorkování je pro každou hodnotu vyhrazen jeden byte (Obrázek 3.9), při dvanáctibitovém jsou pro každé dvě hodnoty vyhrazeny 3 byty (viz Obrázek 3.10). Do prvního je uloženo 8 nejvýznamnějších bitů první hodnoty, do druhého na nejvýznamnější bity zbylé čtyři bity první hodnoty a na zbylé bity čtyři nejvýznamnější bity druhé hodnoty a do třetího zbývajících osm bitů druhé hodnoty.



Obrázek 3.9 – Ukázka struktury datového paketu při osmibitovém vzorkování pěti kanálů



Obrázek 3.10 – Ukázka struktury datového paketu při dvanáctibitovém vzorkování pěti kanálů

3.2.3 Implementace

Při implementaci firmwaru pro mikrokontrolér, implementaci obslužného programu a vytváření ovladače zařízení jsem vycházel z návodu [14], který ukazuje, jak je možné vytvořit grafické uživatelské rozhraní (GUI) pro USB komunikaci s JM60. Na stránce tohoto návodu jsou volně ke stažení i použité zdrojové kódy, jejich součástí jsou i knihovny s funkcemi pro komunikaci přes rozhraní USB. Tyto knihovny jsem použil při implementaci, čímž jsem komunikaci přesunul na vyšší úroveň abstrakce. Obsahují totiž funkce na inicializaci USB modulu, odesílání a příjem USB paketů a obsluhu endpointu 0. Stejně jako v návodu je zařízení třídy *vendor-specific* (zařízení specifické výrobcem) a je k němu tedy potřeba vytvořit i ovladač (kapitola 3.3).

Na přiloženém CD (Příloha 1) je firmware vytvořený v programu *CodeWarrior for Microcontrollers V6.1* ve složce `Firmware/`, zdrojové kódy ve složce `Firmware/Sources`. Ke komunikaci přes USB jsou podstatné následující soubory. Jedná se právě o soubory převzaté z [14], některé z nich však bylo nutné modifikovat:

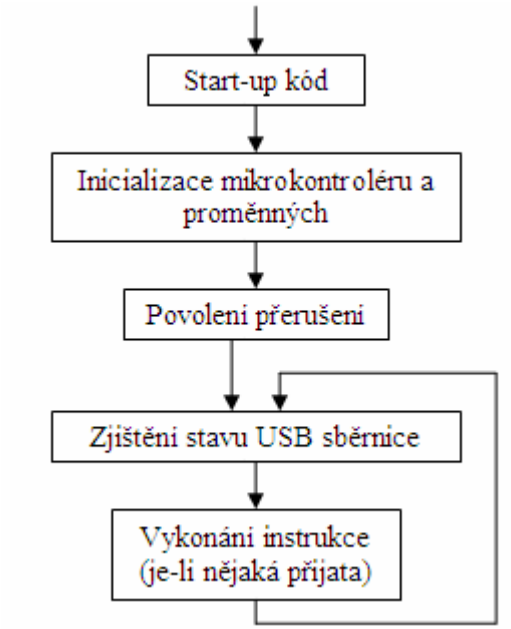
- `typedef.h` – Obsahuje definice použitých datových typů. Mezi ně jsem přidal definici typu *UINT8* (bezznaménkové celé číslo uložené na 8 bitech), který jsem využíval ve funkcích.
- `USB_User_API.c` – Implementované funkce pro inicializaci prvního a druhého endpointu, příjem a odeslání USB paketů a pro zjištění, jestli nepřišel nějaký paket na endpoint typu OUT. Původnímu soubor jsem modifikoval pro dva použité endpointy (z původních 4) a funkce optimalizoval pro použitou komunikaci.
- `USB_User_API.h` – Hlavičkový soubor pro `USB_User_API.c`. Z původního souboru jsem odebral některé deklarace proměnných z důvodu změny počtu endpointů.
- `Usb_Drv/Usb_Bdt.h` – Hlavičkový soubor s definicemi struktur pro tabulky buffer deskriptorů (kapitola 2.2.3). Soubor použit beze změny.

- `Usb_Drv/Usb_Config.h` – Hlavičkový soubor s definicemi hodnot používaných při obsluze endpointu 0. Soubor použit beze změny.
- `Usb_Drv/Usb_Descriptor.c` – Soubor s device, configuration, endpoint, interface a string deskriptory. Device deskriptor obsahuje mj. VID (0x15A2 – fa. Freescale), PID (0x0050) a typ zařízení (vendor-specific). V configuration deskriptoru je nastaven např. maximální odebíraný proud zařízení z USB (100mA). Vlastnosti endpointů - EP1 typu interrupt OUT s intervalem 32 ms a EP2 typu interrupt IN s intervalem 1 ms (podle 3.2.1 – jsou uloženy v endpoint deskriptorech. Do string deskriptoru jsem uložil řeťězce: *„JM60 Channel Monitor“* a *„xnevor02, FIT VUT Brno 2013“*.
- `Usb_Drv/Usb_Descriptor.h` – Hlavičkový soubor pro `USB_Descriptor.c`. Obsahuje definice struktur deskriptorů, deklarace použitých proměnných s deskriptory a definice některých hodnot daných standardem USB, které jsou v deskriptorech použity. Bylo nutné změnit velikost bufferů obou požitých endpointů a obsah struktury *USB_CFG* (snížen počet endpoint descriptorů).
- `Usb_Drv/Usb_Drv.c` – Obsahuje funkce pro řízení USB modulu JM60. Soubor použit beze změny.
- `Usb_Drv/Usb_Drv.h` – Hlavičkový soubor pro `USB_Drv.c`, použit beze změny.
- `Usb_Drv/Usb_Ep0_Handler.c` – Obsahuje funkce pro obsluhu obousměrného řídícího endpointu 0. Soubor použit beze změny.
- `Usb_Drv/Usb_Ep0_Handler.h` – Hlavičkový soubor pro `USB_Ep0_Handler.c`. V komentáři jeho hlavičce jsem opravil špatně uvedené jméno souboru.

Za pomoci těchto souborů stačí inicializovat USB modul mikrokontroléru funkcí `Initialize_USBModule` (soubor Usb_Drv.c) a použít následující funkce:

- `Check_USBBus_Status` (soubor Usb_Drv.c) – Zjišťuje, jestli je modul připojen na USB sběrnici.
- `CheckEndPointOUT` (soubor USB_User_API.c) – Pokud přisel nějaký paket na endpoint 1 (parametr funkce musí být číslo „1“) typu OUT, vrátí hodnotu jedna. Paket je pak uložen v poli `UINT8 EP1_Buffer[8]`. V opačném případě vrací nulu.
- `EndPoint_IN` (soubor USB_User_API.c) – Odešle paket, který je uložen v poli `UINT8 EP2_Buffer[64]` přes endpoint 2. První parametr funkce musí být číslo endpointu (2), druhý velikost paketu v bytech.

Po startu mikrokontroléru a provedení tzv. „start-up“ kódu (Start08.c) vygenerovaného programem *CodeWarrior* je volána funkce `main` (main.c). Po inicializaci používaných proměnných, registrů mikrokontroléru (funkcí `Init_Sys`) jsou povolena přerušení a program se ocitá v nekonečné smyčce (Obrázek 3.11).



Obrázek 3.11 – Vývojový diagram mikrokontroléru

Inicializace mikrokontroléru

Funkcí `Init_Sys` (init.c) se inicializují registry, které ovlivňují chování mikroprocesoru a jeho periferií. Jsou z ní postupně volání funkce ze stejného souboru (pokud není uvedeno jinak):

- `Mcu_Init` – Inicializace MCU.
- `MCG_Init` – Inicializace MCG modulu (stejně jako inicializace MCU je převzata z návodu [14]). Za pomoci externího 12MHz krystalu (viz prototyp zařízení – kapitola 3.1) bude modul generovat 24MHz hodinový signál.
- `GPIO_Init` – Inicializace vstupně/výstupních pinů. Pin, na který je v zařízení zapojena LED dioda D1, je deklarován jako výstupní a dioda je zhasnuta.
- `ADC_Init` – Inicializace A/D převodníku (A/D převodník viz 2.2.3). Pomocí registrů `APCTL1` a `APCTL2` je všech 12 kanálů, které mají vývod na některý z pinů mikrokontroléru, nastaveno jako vstup A/D převodníku. Ostatní registry jsou nastaveny tak, aby vzorkování trvalo co nejkratší možnou dobu a maximální vzorkovací frekvence mohla být co nejvyšší.
- `Rtc_Init` – Inicializace RTC (RTC viz 2.2.3). Jako zdroj hodin je nastaven vnitřní 1kHz oscilátor a přerušení od RTC modulu jsou zakázána. Hodinový signál nebude nijak upravován, modul tedy bude čítat po milisekundách.
- `Initialize_USBModule` (soubor Usb_Drv.c) – Inicializace USB modulu, zmíněna výše.

Nekonečná smyčka

V každém cyklu nekonečné smyčky jsou volány funkce `Check_USBBus_Status` a `CheckEndPointOUT`. V případě, že druhá funkce zjistí, že USB modul obdržel paket na endpointu 1, chová se program podle přijaté instrukce:

- **Začni vzorkovat** – Pokud už nějaké vzorkování probíhá, je zastaveno. Proměnné v programu jsou upraveny tak, aby byly vzorkovány kanály, které se mají podle instrukce vzorkovat

(funkce `channelsUpdate`). Je nastaven vzorkovací interval (funkcí `rtcIntervalSet`) jako časový interval, po kterém má RTC vyvolávat přerušení. Funkce `adcBitDepthSet` zajistí nastavení správné bitové hloubky A/D převodníku změnou registru `ADCCFG`.

Podle počtu vzorkovaných kanálů a hloubky vzorkování je vypočtena velikost, kterou zaberou hodnoty jedné skupiny měření. Z ní, z intervalu vzorkování a minimálního počtu USB paketů za sekundu je pak dopočítán počet skupin měření ukládaných do jednoho paketu (funkce `samplesPerPacketUpdate`). Pokud by musely být pro dodržení minimálního počtu USB paketů za sekundu odesílány prázdné pakety (frekvence vzorkování je nižší než požadovaný počet paketů), je do paketu ukládána právě jedna skupina měření a na minimální počet USB paketů není brán zřetel.

Určený počet skupin měření v paketu je společně s ID přenosu uložen do prvních dvou bytů v odesílaných paketech (viz struktura paketů 3.2.2). LED dioda oznamující probíhající vzorkování (D1) je rozsvícena. Pokud nemá být vzorkován žádný kanál, je odeslán jediný paket s daným ID přenosu bez jakýchkoli naměřených hodnot, v opačném případě je povoleno přerušení RTC modulu.

- **Ukonči vzorkování** – Jsou zakázány přerušení od RTC modulu a LED dioda připojená k mikrokontroléru zhasne.

- **Změň vzorkované kanály** – Pokud žádné vzorkování neprobíhá, ztrácí instrukce smysl a nic se neprovede. V opačném případě je chování podobné jako u instrukce *Začni vzorkovat*, pouze není nastavována bitová hloubka a interval vzorkování (oboje zůstává zachováno).

Samotné vzorkování (jedné skupiny měření) probíhá v obsluze přerušení RTC modulu. Postupně jsou prováděny A/D konverze všech sledovaných kanálů (od kanálu s nejnižším číslem po kanál s nejvyšším) a převedená čísla jsou ukládána na patřičné pozice v paketu. Konec A/D konverze je zjištěn dotazováním (tzv. „polling“) nad bitem *COCO* v registru `ADCSC1`. Pokud je A/D konverze dokončena a v paketu jsou uloženy všechny skupiny měření (jejich počet známe z funkce `samplesPerPacketUpdate`), je paket odeslán.

3.3 Program komunikující s JM60

Při implementaci USB komunikace na straně počítače jsem vycházel ze stejného návodu jako při implementaci firmwaru ([14]).

3.3.1 Ovladač zařízení

Vyrobené zařízení je třídy *vendor-specific* (zařízení specifické výrobcem) a je k němu potřeba vytvořit ovladač. Jednou z možností je implementace vlastního ovladače, což není triviální úkol a není ani tématem práce, jinou je využít generický ovladač *WinUSB*. Ten umožňuje všechny USB přenosy kromě isochronního a je podporován operačním systémem Windows od verze XP SP2 [15].

K vygenerování ovladače založeného na *WinUSB* jsem použil program *WinUSB INF generátor* firmy Freescale (ke stažení také v návodu [14]). Vyplnil jsem VID (0x15A2), PID (0x0050) a tzv. „Device release number“ (0x0100) na základě deskriptorů ve firmwaru. Dále jsem uvedl jméno výrobce (“xnevor02 @ FIT VUT Brno 2013“), třídu zařízení (“JM60 devices“), jeho jméno (“JM60

Channel Monitor“) a program vygeneroval ovladač zařízení, který je k nalezení ve složce `Driver/` na přiloženém CD (Příloha 1), a GUID (globally unique identifier) zařízení: d7217eac-de67-42ba-bf9b-c4169585b93a.

3.3.2 SimpleUSB.dll

SimpleUSB.dll je knihovna firmy Freescale pro programy vytvářené v jazyce C#, která je volně ke stažení na stránce [14]. Umožňuje jednoduchou komunikaci s USB zařízením na úrovni endpointů.

Pro komunikaci se zařízením je třeba přiřadit jeho GUID do stejnojmenné „property“ objektu třídy SimpleUSB (vytvořeného bezparametrickým konstruktorem) a zavolat metodu `OpenConnection`. Po úspěšném spojení je vyvolána událost `onDeviceConnect`. V tuto chvíli je možné pomocí metody `WriteData` posílat data na endpointy s přenosem bulk nebo interrupt. Z bulk endpointu se čte metodou `ReadBulkEndpoint`. `StartReadingInterruptEndPoint` vytváří nové vlákno, které přijímá data z jednoho endpointu typu interrupt, úspěšný paket ohlásí událostí `onReadComplete` a k datům se programátor dostane pomocí metody `ReadInterruptEndpoint`. Vlákno končí metoda `StopReadingInterruptEndPoint`. V případě, že je zařízení odpojeno, je vyvolána událost `onDeviceDisconnected`. Spojení lze ukončit metodou `CloseConnection`. Celá dokumentace ke knihovně je ke stažení zde [16].

3.3.3 Implementace

Pro počítačovou aplikaci, která má demonstrovat funkčnost zařízení, sbírat hodnoty vzorkovaných kanálů a vizualizovat jejich průběhy v čase, využiji nabízenou knihovnu SimpleUSB.dll a budu tedy aplikaci implementovat v programovacím jazyce C#. Dále popisovanou aplikaci vyvíjenou v prostředí *Microsoft Visual Studio 2010 Express* na platformě *Microsoft .NET Framework 4* a její zdrojové kódy je možné nalézt na přiloženém CD (Příloha 1) ve složce `GUI/`.

Vytvořil jsem 4 třídy ve jmenném prostoru „USB_GUI“: `MainWindow`, `SettingsWindow`, `Settings` a `Transfer`.

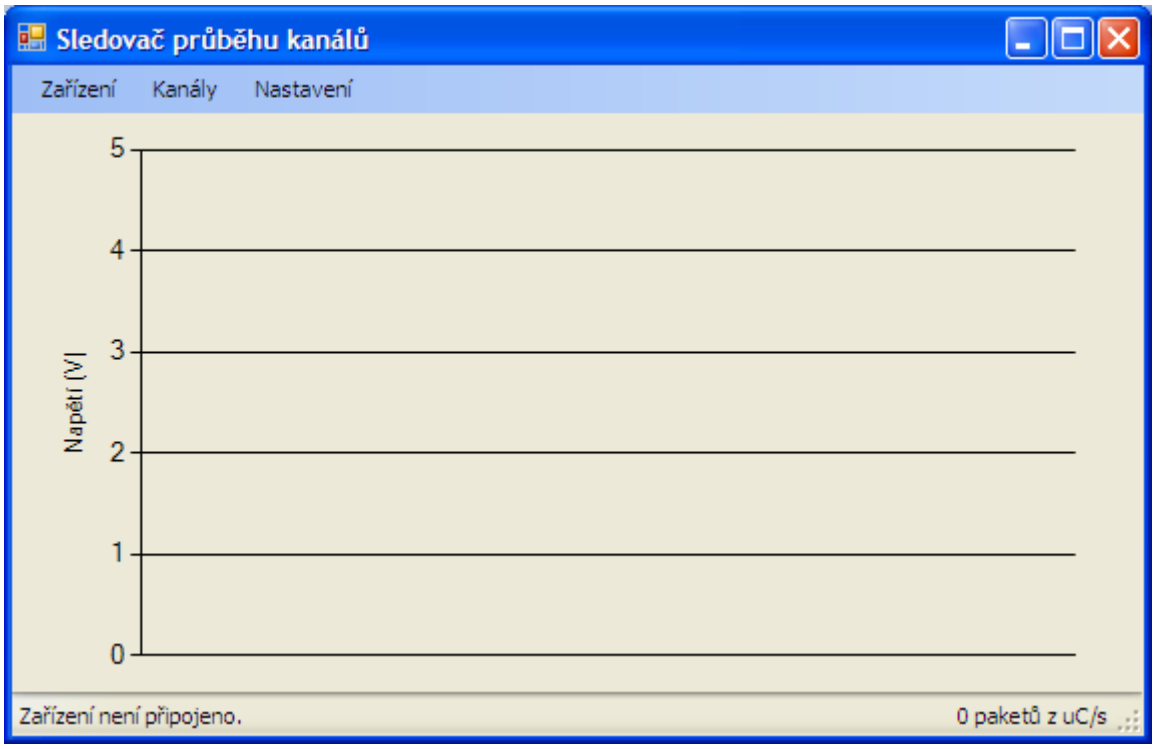
- **MainWindow** – Třída využívající API rozhraní *Windows Forms*. Představuje hlavní okno, které má kromě plochy pro vizualizaci také programové menu a stavový řádek (Obrázek 3.12). V menu je možné otevřít či ukončit komunikaci se zařízením (otevřena je automaticky), vybrat kanály, které se mají sledovat a spustit okno s nastavením programu. Stavový řádek obsahuje na pravé straně počet příchozích paketů ze zařízení za sekundu a stav zařízení v textové podobě na pravé straně. O implementaci třídy dále.

- **SettingWindow** – Třída využívající API rozhraní *Windows Forms*. Představuje okno *Nastavení* viz Obrázek 3.13, ve kterém lze nastavovat parametry vzorkování sledovaných kanálů, vlastnosti grafu, počty bodů v grafu a kalibrovat hodnoty z A/D převodníku.

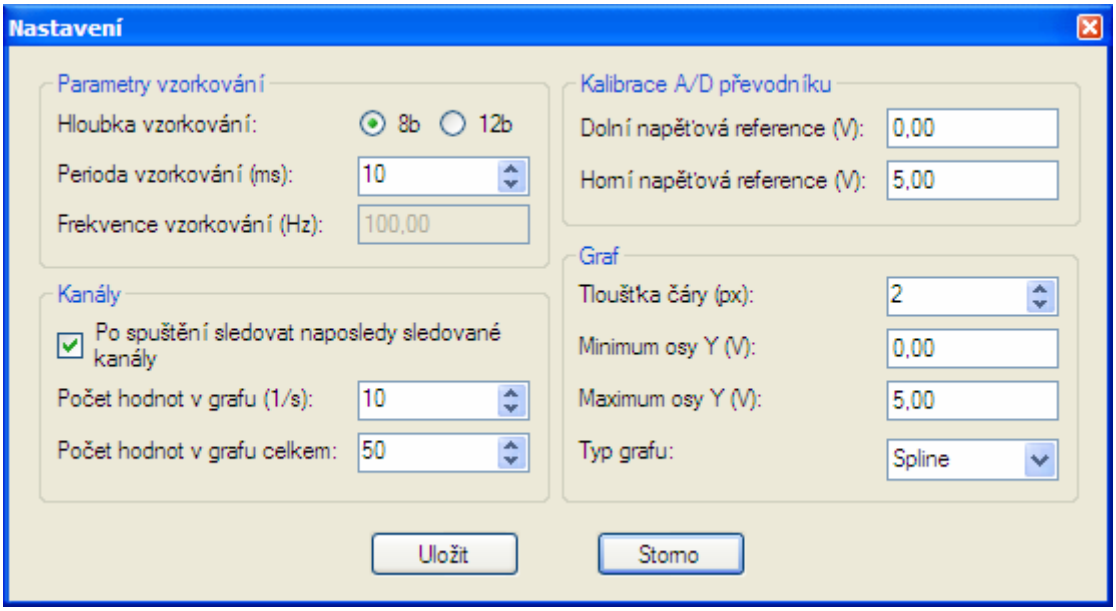
  Pro ukládání programového nastavení jsem zvolil *Settings.settings*, takže se nemusím starat o ukládání na disk a načítání. Mezi takto ukládané hodnoty patří i údaje nastavované na obrázku. Po vytvoření instance této třídy jsou do formulářových prvků v GUI vloženy hodnoty načtené ze *Settings*.

  Při úpravě čtyř polí pro zadávání desetinných čísel jsou hodnoty průběžně kontrolovány a případně červeně podbarveny. Kromě platného zápisu čísel musí být

napěťové reference různé a minimum zobrazované na osy Y v grafu menší než maximum. Ostatní prvky neumožňují zadat špatné hodnoty, proto nejsou podbarvovány. *Perioda vzorkování* je omezena rozsahem 1 – 255 ms, *počet hodnot v grafu* (určuje minimální počet USB paketů ze zařízení) rozsahem 1 – 100 na kanál za sekundu, *počet hodnot v grafu celkem* rozsahem 10 – 1000 a *tloušťka čáry* minimem 1 px a maximem 10 px.



Obrázek 3.12 – Hlavní okno aplikace



Obrázek 3.13 – Okno *Nastavení*

Při stisku tlačítka *Uložit* se ze všech hodnot vytvoří objekt třídy *Settings*, který je předán objektu třídy *MainWindow*, který toto okno vytvořil. Ten upraví chování aplikace

podle provedených změn a uloží hodnoty zpět do *Settings.settings*. Při stisku tlačítka *Cancel* nebo zavření okna se změny neukládají.

- **Settings** – Třída reprezentující nastavení aplikace. Při volání bezparametrického konstruktoru obsahuje defaultní nastavení, při volání druhého naimplementovaného konstruktoru jsou jako parametry předávané všechny nastavované hodnoty. Třída má metodu *Save*, která uloží hodnoty do *Settings.settings*.

- **Transfer** – Třída reprezentující datové streamy ze zařízení. Její objekt uchovává informace o vzorkování s jedním *ID přenosu* – hloubku vzorkování, vzorkované kanály, velikost hodnot skupiny měření v bytech a ID přenosu.

Pro vizualizaci naměřených hodnot jsem použil graf od firmy Microsoft, objekt třídy `System.Windows.Forms.DataVisualization.Charting.Chart`. V konstruktoru třídy *MainWindow* jsou po inicializaci komponent (metodou `InitializeComponent` generovanou designérem *Microsoft Visual Studia*) vytvořeny položky do menu pod záložku „Kanály“ a řady do grafu pro počet kanálů uložený v *Settings.settings* (12). Otevře se spojení s mikrokontrolérem (funkce `OpenConnection` knihovny *SimpleUSB.dll*) a začne obnovování grafu.

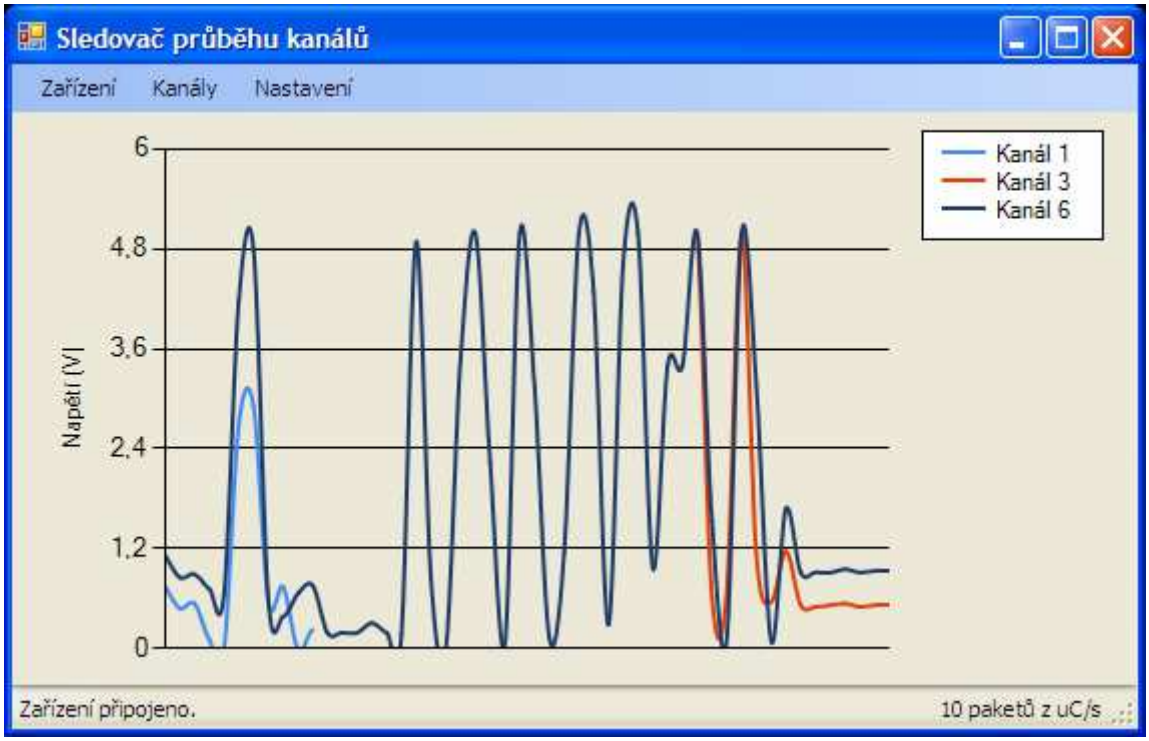
Pokud se úspěšně naváže spojení s mikrokontrolérem, vygeneruje se ID přenosu, vytvoří nový objekt třídy Transfer, odešle na endpoint 1 zařízení instrukci *Začni vzorkovat* s informacemi o vzorkování nastavenými v programu a spustí se čtení z endpointu 2 v novém vlákně.

Když přijde datový paket, zjistí se z něj nejdříve ID přenosu. Pokud je to už paket s novým ID, je potřeba aktualizovat kanály, do kterých se budou hodnoty z paketů ukládat, neboť se mohly od minulého přenosu změnit. Naměřené hodnoty jsou upravené pomocí kalibračních hodnot zadaných v nastavení programu a uloženy v datovém typu *double* do seznamů naměřených hodnot v poli těchto seznamů `List<double>[] channelValues`. Indexování v poli je pomocí čísla kanálu zmenšeného o jedničku – tedy od 0 do 11.

Po spuštění programu jsou sledovány stejné kanály, které byly sledovány před posledním ukončením programu, pokud je tato vlastnost v nastavení programu zatržená. Změní-li uživatel sledované kanály – pomocí menu nebo klávesových zkratek F1 – F12 a probíhá-li tou dobou sledování, dojde k vygenerování nového ID přenosu, vytvoření nového objektu třídy Transfer a odeslání instrukce *Změň vzorkované kanály*.

Když uživatel stiskne tlačítko *Uložit* v okně *Nastavení* a probíhá v té době vzorkování, odešle se zařízení instrukce *Ukonči vzorkování*, uloží se nastavení programu a provedou se patřičné změny. Poté je vygenerováno nové ID přenosu, vytvořen nový objekt třídy Transfer a odešle se instrukce *Začni vzorkovat*. Pokud vzorkování neprobíhá, stačí uložit nastavení programu a provést s tím spojené změny.

Při počáteční inicializaci je do každé řady grafu (jedna řada odpovídá jednomu kanálu) umístěno tolik bodů, kolik jich tam má být podle nastavení. Všechny body jsou ale prázdné a na grafu není nic vidět. O překreslování grafu se stará časovač, který po intervalech vypočítaných z *počtu hodnot v grafu na kanál za sekundu* volá funkci `GraphActualize`. Ta z každé řady grafu odebere bod s nejstarší hodnotou kanálu a přidá na konec bod nový. Pokud není zařízení připojeno nebo se kanál nevzorkuje, je bod prázdný a v grafu se nezobrazuje, v opačném případě se vloží poslední přijatá naměřená hodnota od mikrokontroléru – z pole seznamů `channelValues`. To způsobuje, že křivky z naměřených hodnot již nesledovaných kanálů na grafu „dojíždějí“ – viz Obrázek 3.14.



Obrázek 3.14 – Ukázka činnosti aplikace

O aktualizování údaje s *počty přijatých USB paketů za sekundu* ve stavovém řádku aplikace se stará jiný časovač. Jednou za vteřinu tento údaj zaktualizuje a resetuje čítač paketů – proměnnou typu `int`.

Při změně velikosti okna je automaticky změněna i velikost grafu. Pokud uživatel ukončí spojení se zařízením (pomocí menu) nebo zavře aplikaci, odešle se ještě zařízení instrukce *Ukonči vzorkování*.

3.3.4 Problémy s knihovnou *SimpleUSB.dll*

Při implementaci jsem objevil chyby, které ukrývá knihovna *SimpleUSB.dll*:

1. Pokud je otevřeno metodou `OpenConnection` spojení se zařízením, čte se z interrupt endpointu metodou `StartReadingInterruptEndPoint` a zařízení přestane komunikovat přes USB s počítačem – např. je fyzicky odpojeno od sběrnice USB nebo manuálně tlačítkem S1 resetováno, třída *SimpleUSB* generuje jednu událost `onReadComplete` za druhou.

   Řešení: Na začátku obslužné rutiny (metody) pro událost `onReadComplete` se kontroluje, zdali je zařízení připojeno. Pokud není, zobrazí se uživateli varovné hlášení o odpojení zařízení. Dokud uživatel nestiskne tlačítko *Opakovat* nebo *Storno*, kterým se aplikace ukončuje, nebyla dokončena tato rutina, proto není volána tato metoda pro další stejnojmennou událost a vytížení procesoru není díky této aplikaci 100%.

2. Při používání aplikace se může ve výjimečném případě – mnohonásobná velmi rychlá změna sledovaných kanálů – stát, že aplikace přestane odpovídat.

Řešení: Reset zařízení tlačítkem S1.

3.4 Kroky k inovaci KITu

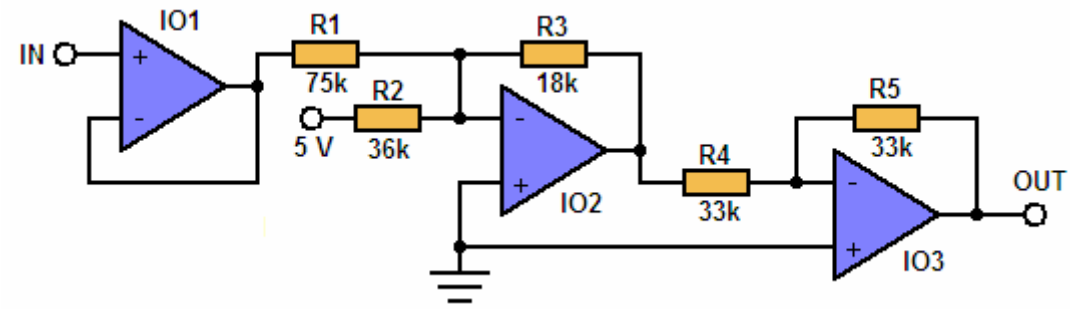
Takto vytvořené zařízení rozšiřuje možnosti sledování napětí oproti A/D převodníku laboratorního přípravku, takže by jej bylo možné zakomponovat do KITu při jeho inovaci.

KIT obsahuje stabilizované napětí 5 V, kterým by mohlo být zařízení napájeno. Přesnost měření je odvozená od správné kalibrace napětí V_{REFL} vůči nulovému potenciálu a napětí přiváděné přes USB kabel může kolísat a na různých počítačích se dokonce může mírně lišit.

Interval měřeného napětí A/D převodníkem se pohybuje od 0 V do napětí na pinu V_{REFL}, které může být podle katalogového listu [4] od 2,7 V do napájecího napětí mikrokontroléru – tedy 5 V. Minimální rozsah napětí, který potřebujeme při používání KITu měřit, činí -10 V až +10V. Je tedy nutné u každého používaného kanálu A/D převodníku vložit mezi konektor CON3 a měřené napětí nějaký obvod, který lineárně převede alespoň interval vstupního napětí -10 V až 10 V na interval výstupního napětí 0 V až 5 V. Ten může být vytvořen pomocí operačních zesilovačů.

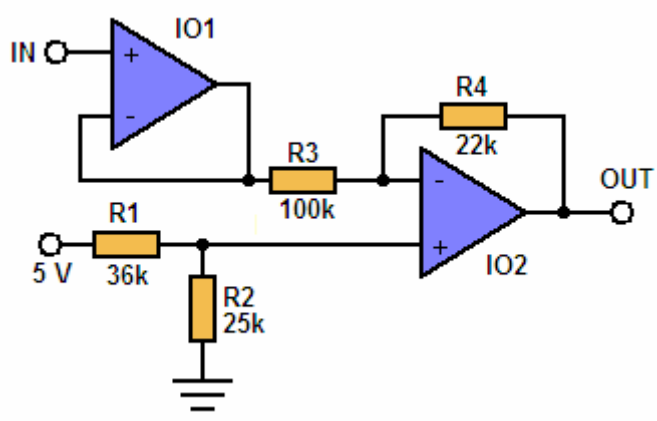
3.4.1 Obvod pro transformaci napětí

Pro převod napětí je potřeba vstupní napětí vydělit alespoň čtyřmi a přičíst k němu polovinu napětí na pinu V_{REFL} - tedy 2,5 V. To by mohl řešit obvod viz Obrázek 3.15. Operační zesilovač IO1 slouží jako sledovač napětí, na jeho výstupu zůstává stejné napětí jako na svorce IN. IO2 násobí toto napětí hodnotou -0,24 (podíl -R3/R1) a přičítá k němu napětí -2,5 V (-R3/R2 * 5 V). Operační zesilovač IO3 slouží jako invertor se zesílením -1 (podíl -R5/R4). Na napětí 0 V až 5 V je tak převeden přibližně rozsah -10,4 V až 10,4 V. Odpory mají svoji toleranci a OZ nejsou ideální, takže je tento rozsah potřeba přesně změřit a jeho meze uvést jako dolní a horní napěťovou referenci v nastavení aplikace.



Obrázek 3.15 – Příklad obvodu na transformaci napětí č. 1

K transformaci napětí by mohl sloužit i obvod viz Obrázek 3.16. První OZ je opět zapojen jako sledovač napětí, druhý vytváří na svém výstupu napětí, které je dáno součtem 2,5 V (podle vztahu: R2/(R1+R2) * (R3+R4)/R3 * 5 V) a -0,22násobku (podíl -R4/R5) napětí na svorce IN. Na napětí 0 V až 5 V je tak převeden přibližně rozsah 11,4 V až -11,4 V (POZOR! Nikoliv -11,4 V až 11,4 V). Tomu jsem uzpůsobil i aplikaci, do které může uživatel zadat dolní napěťovou referenci převáděného napětí i vyšší než horní. Tento vypočtený rozsah je opět pouze teoretický, závisí na reálných použitých součástkách.



Obrázek 3.16 – Příklad obvodu na transformaci napětí č. 2

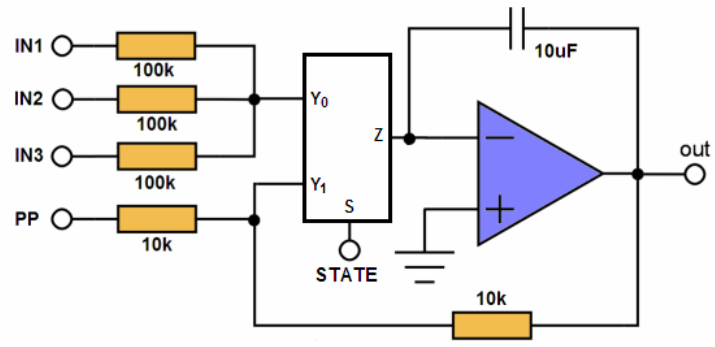
Operační zesilovač IO1 zapojený jako sledovač napětí jsem předřadil v obou zapojeních ostatním kvůli svému teoreticky nekonečnému vstupnímu odporu. Ten se očekává od každého sledujícího zařízení včetně voltmetru. Pokud bychom IO1 vypustili a svorku IN zapojili na jeho výstup, byl by vstupní odpor obvodu roven hodnotě odporu R1 (resp. R3 pro druhý obvod), neboť operační zesilovače zachovávají na svém invertujícím vstupu nulový potenciál.

3.4.2 Možná rozšíření

Při inovaci KITu můžeme využít zařízení i k jiným účelům než jen ke sledování napětí na kanálech. Mikrokontrolér má v prototypu zařízení mnoho nevyužitých periferií a pinů. Některé příklady dalšího využití ukazuje tato kapitola.

Přepínání stavů integrátorů

Počítačovou aplikaci můžeme doplnit tlačítkem/y pro změnu stavu integrátorů. K tomuto účelu je potřeba implementovat pro USB komunikaci na endpointu 1 novou instrukci. Na straně mikrokontroléru by mohl její příjem způsobit změnu hodnoty na jednom z jeho výstupních pinů. Ten by ovládal např. digitálně řízené přepínače (jednobitové analogové multiplexory) podle schématu vodičem STATE viz Obrázek 3.17. Tyto přepínače vyrábí např. firma *Analog Devices, Inc.* [17].



Obrázek 3.17 – Ovládání stavu integrátoru analogovým multiplexorem

Pokud by bylo v rámci inovací sníženo symetrické napětí ±10 V na ±7,5 V, mohl by se pro přepínání stavu integrátorů použít integrovaný obvod *4053*. Ten obsahuje tři jednobitové analogové multiplexory a stačil by pro celý KIT jediný [18].

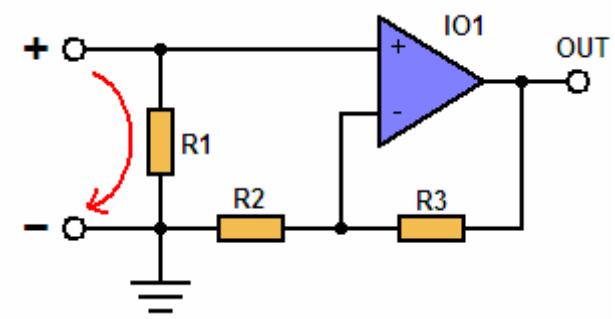
Generování obdélníkových signálů

Modul TPM mikrokontroléru JM60 má univerzální časovací kanály, které umožňují pulzně šířkovou modulaci. Dva z těchto kanálů jsou napojeny na konektor CON4. Přidáním další instrukce do USB komunikace bychom mohli zapínat a vypínat modulátory a nastavovat periodu a třídu obdélníkových signálů.

Ampérmetr

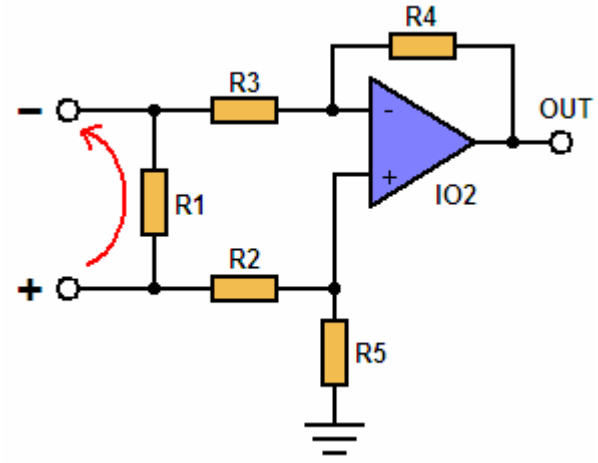
Abych mohl měřit elektrický proud pomocí prototypu zařízení, musím vytvořit nějaký obvod, který mi umožní nejprve proud lineárně převést na napětí a teprve to potom měřit pomocí zařízení. Nejjednodušší variantou je vložit do cesty proudu rezistor s malým odporem, protože ampérmetr má mít teoreticky nulový vstupní odpor, a napětí na něm zesílit.

Pokud je možné záporný vývod ampérmetru v takovémto obvodu uzemnit a měřit tak proud proti zemi (Obrázek 3.18), stačí přivést kladný vývod na neinvertující vstup OZ a odpory R2 a R3 určují zesílení (1 + R3/R2) výstupního napětí vůči napětí na odporu R1 (např. 0R1).



Obrázek 3.18 – Obvod pro měření proudu s uzemněným vývodem

Měřený obvod může být s měřícím nějak napěťově svázán a při uzemnění záporného pólu ampérmetru by došlo ke zkratu. V tomto případě využiji v obvodu OZ jako diferenční zesilovač – viz Obrázek 3.19. Proud ampérmetrem je opět měřen pomocí napětí, které vznikne na odporu R1 (např. 0R1). Má-li OZ zesilovač zesilovat pouze toto napětí, musí být poměry odporů R4:R3 a R5:R2 stejné. Zesílení výstupního napětí vůči napětí na R1 pak udává právě tento poměr [19].



Obrázek 3.19 – Obvod pro měření proudu pomocí rozdílového zesilovače

4 Závěr

Podařilo se mi vytvořit zařízení, které komunikuje přes sběrnici USB s počítačovou aplikací. Umí sledovat až 12 kanálů, na kterých měří napětí v rozsahu 0 až 5 V. Přesnost měření závisí na stabilitě napájecího napětí a jeho kalibraci, chybě měření A/D převodníku a jeho vstupním odporu. Při 12tibitové vzorkování je rozlišitelná jednotka přibližně 1,2 mV.

Činnost zařízení je ovládána z počítačového programu, který byl otestován na 32bitových operačních systémech Windows XP SP3 a Windows 7. Vizualizuje průběh napětí na kanálech, které uživatel vybere. Naměřené hodnoty se mapují na napěťový interval zadaný v nastavení aplikace, které se společně s naposledy sledovanými kanály uchovává na počítači i po zavření programu.

Kromě vlastností vykreslujícího grafu se dá v aplikaci nastavit i perioda vzorkování (v ms). Její přesnost závisí na frekvenci vnitřního 1kHz zdroje hodinové signálu, která se může mezi jednotlivými kusy JM60 pohybovat od 900 Hz po 1500 Hz [4]. V mikrokontroléru použitém v zařízení jsem naměřil přibližně 1040 Hz. Maximální vzorkovací frekvence je omezena rychlostí A/D převodu, taktovací frekvencí CPU a implementací firmwaru. S 32kHz zdrojem hodinového signálu pro čítač reálného času se mi podařilo vzorkovat 1 kanál frekvencí až 16 kHz nebo 3 kanály frekvencí až 8kHz a naměřené hodnoty současně odesílat přes USB.

Po umístění do krabičky můžeme zařízení používat samostatně, můžeme ho zakomponovat i do laboratorního přípravku IPR při jeho inovaci, neboť rozšiřuje možnosti sledování napětí oproti stávajícímu A/D převodníku. Pro tento případ jsem navrhnul doplňující elektrické obvody, které by umožnily měřit pomocí zařízení i proud, napěťové rozsahy používané v KITu nebo využít mikrokontrolér pro přepínání stavu integrátorů. Zařízení by pak navíc mohlo sloužit i ke generování obdélníkového signálu s nastavitelnou periodou a střídou.

V rámci dalšího vývoje by mohla být k zařízení navržena další rozšíření, vyřešena ochrana A/D převodníku proti velkému napětí na vstupních kanálech. Do počítačové aplikace bylo vhodné doplnit funkcí pro uložení naměřených hodnot z proměnné `channelValues` do souboru a přidat vlákno pro zpracování přijatých paketů. S dostupnými knihovnami pro firmware mikrokontroléru by ho bylo možné upravit tak, aby bylo zařízení považováno počítačem jako zařízení třídy HID a nebyly k němu potřeba dodávat ovladače, neboť jsou součástí většiny moderních operačních systémů.

Literatura

[1] STRNADEL, Josef. Laboratoře předmětu IPR (Zadání) - lab. č. 1. *Stránka laboratoří předmětu Prvky počítačů (IPR)* [online]. 2011 [cit. 2013-04-26]. Dostupné z: http://www.fit.vutbr.cz/~strnadel/ipr/ipr_lab01.htm

[2] STRNADEL, Josef. Stránka laboratoří předmětu Prvky počítačů (IPR). *Stránka laboratoří předmětu Prvky počítačů (IPR)* [online]. 2011 [cit. 2013-04-26]. Dostupné z: http://www.fit.vutbr.cz/~strnadel/ipr/

[3] SMETANA, Ctirad. *Praktická elektroakustika*. 1. vyd. Praha: SNTL, 1981, 696 s.

[4] FREESCALE SEMICONDUCTOR. *MC9S08JM60 Series Data Sheet* [online]. 2009 [cit. 2013-01-22]. Dostupné z: http://cache.freescale.com/files/microcontrollers/doc/data_sheet/MC9S08JM60.pdf

[5] LIU, Derek. FREESCALE SEMICONDUCTOR. *USB Device Development with the MC9S08JM60: In-depth Understanding of the MC9S08JM60 USB Module* [online]. 2008 [cit. 2013-01-22]. Dostupné z: http://cache.freescale.com/files/microcontrollers/doc/app_note/AN3560.pdf

[6] Konektor MLW06G. *GM electronic* [online]. 2013 [cit. 2013-05-10]. Dostupné z: http://www.gme.cz/konektory-pro-ploche-kabely-do-dps/konektor-mlw06g-p800-083/

[7] Flexis JM Demonstration Board. *Freescale Semiconductor* [online]. 2013 [cit. 2013-05-10]. Dostupné z: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=DEMOJM

[8] BIDLO, Michal. *Úvod do systému USB mikrokontroléru MC9S08JM60: Demonstrační cvičení IMP* [online]. 2010 [cit. 2013-01-27]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IMP-IT/excs/demo5/IMP_demo_USB.pdf?cid=8661

[9] PEACOCK, Craig. Chapter 4 - Endpoint Types. *USB in a NutShell* [online]. 2010 [cit. 2013-04-29]. Dostupné z: www.beyondlogic.org/usbnutshell/usb4.shtml

[10] Part 3 - Data Flow. MQP ELECTRONICS LTD. *USB Made Simple* [online]. 2008 [cit. 2013-04-29]. Dostupné z: http://www.usbmadesimple.co.uk/ums_3.htm

[11] How to transfer data to USB isochronous endpoints. MICROSOFT. *Microsoft Developer Network* [online]. 2013 [cit. 2013-04-29]. Dostupné z: http://msdn.microsoft.com/en-us/library/windows/hardware/hh406225%28v=vs.85%29.aspx

[12] Prodej USB PID. *ASIX* [online]. 2013 [cit. 2013-05-04]. Dostupné z: http://www.asix.cz/usb_pid.htm

[13] Part 4 - Protocol. MQP ELECTRONICS LTD. *USB Made Simple* [online]. 2008 [cit. 2013-04-29]. Dostupné z: http://www.usbmadesimple.co.uk/ums_4.htm

[14] QUIROZ, Samuel. FREESCALE SEMICONDUCTOR MEXICO. *How to make a Graphical User Interface for your USB application* [online]. 2008 [cit. 2013-01-22]. Dostupné z: https://sites.google.com/site/bitwok/

[15] WinUSB (Windows Drivers). MICROSOFT. *Microsoft Developer Network* [online]. 2013 [cit. 2013-05-06]. Dostupné z: http://msdn.microsoft.com/en-us/library/windows/hardware/ff540196%28v=vs.85%29.aspx

[16] Documentation of SimpleUSB properties and functions [online]. 2013 [cit. 2013-05-06]. Dostupné z: www.freescale.com/files/soft_dev_tools/software_tools/initialization/boot_code_generation/SIMPLE_USB_DOC.zip

[17] Analog Switches. *Analog Devices* [online]. 2013 [cit. 2013-05-07]. Dostupné z: http://www.analog.com/en/switchesmultiplexers/analog-switches/products/index.html

[18] PHILIPS SEMICONDUCTORS. *DATA SHEET HEF4053B MSI* [online]. 1995 [cit. 2013-05-07]. Dostupné z: http://www.gme.cz/dokumentace/427/427-062/dsh.427-062.1.pdf

[19] OPERAČNÍ ZESILOVAČ (OZ) [online]. 2008 [cit. 2013-05-07]. Dostupné z: http://www.outech-havirov.cz/skola/files/knihovna_eltech/ea/oz.pdf

Seznam příloh

Příloha 1. CD

Obsah CD

- ./DPS – Obrázky schématu zařízení, DPS, osazení DPS
  - ./DPS/Eagle – návrh DPS, projekt aplikace *Eagle 6.4.0*
- ./Driver – Ovladač k zařízení
- ./Firmware – Firmware pro JM60, projekt aplikace *CodeWarrior for Microcontrollers V6.1*
- ./GUI – řídící aplikace k zařízení, projekt aplikace *Microsoft Visual Studio 2010 Express*
- ./BP.doc – text bakalářské práce – *Microsoft Word 2003*
- ./BP.docx – text bakalářské práce – *Microsoft Word 2007*
- ./BP.pdf – text bakalářské práce – *Adobe Reader*